AEM Project Structure Deep Dive

Tyler Maynard
AEM DEVELOPER

@TylersDesk www.tylermaynard.com

Overview

Deeper look into the Maven build and project structure

Content-package-maven-plugin

Package filters

AEM specific profiles

Embedding OSGi bundles

What’s Really Happening During a Build

Multiple Modules = Multiple Artifacts

Content packages (AEM packages)

Bundles

Bundles

Java classes and resources

Bundle

OSGi

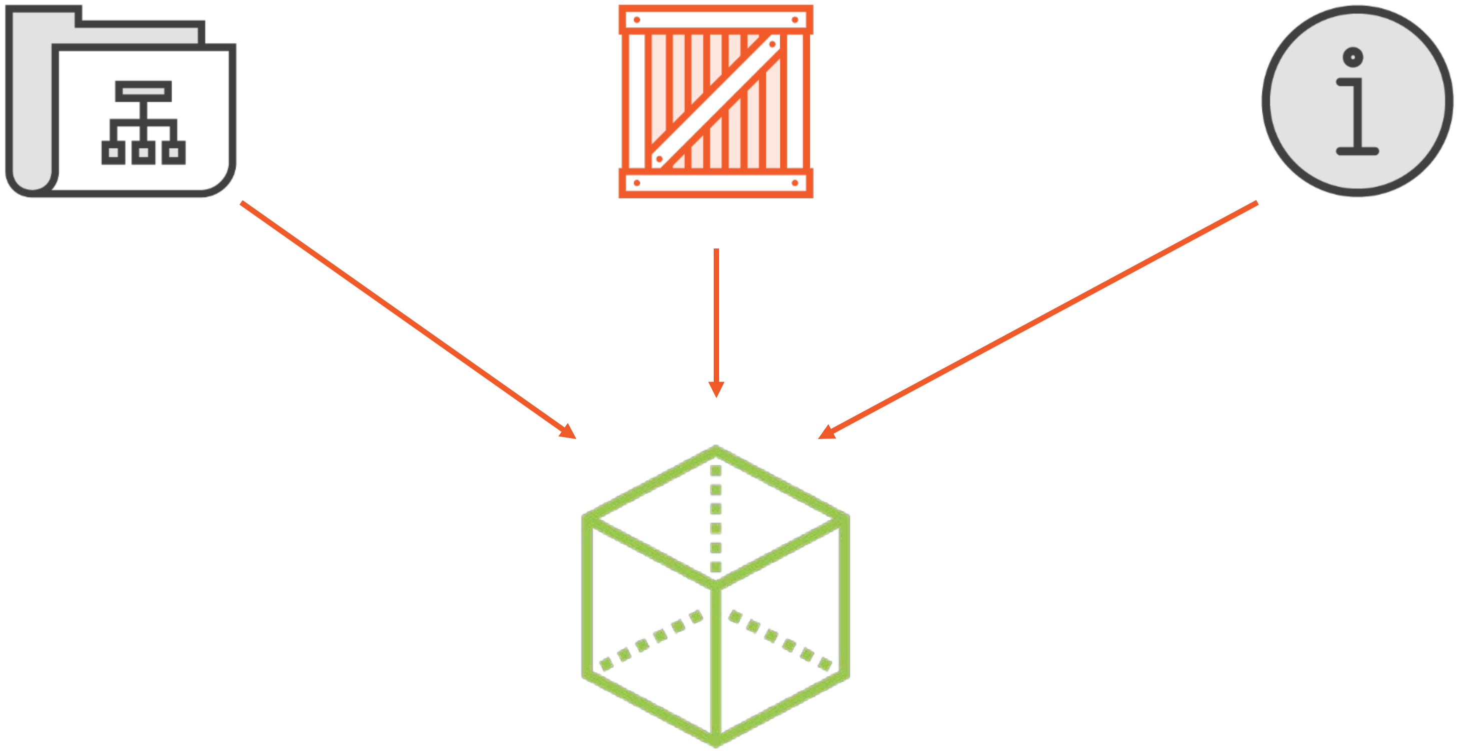
Content Packages

Vault serialized content

Embeds (bundles)

Vault meta information

Package Creation



AEM Package

Maven’s Build Node

```
<build>
   Work Happens Here!
</build>
```

Demo

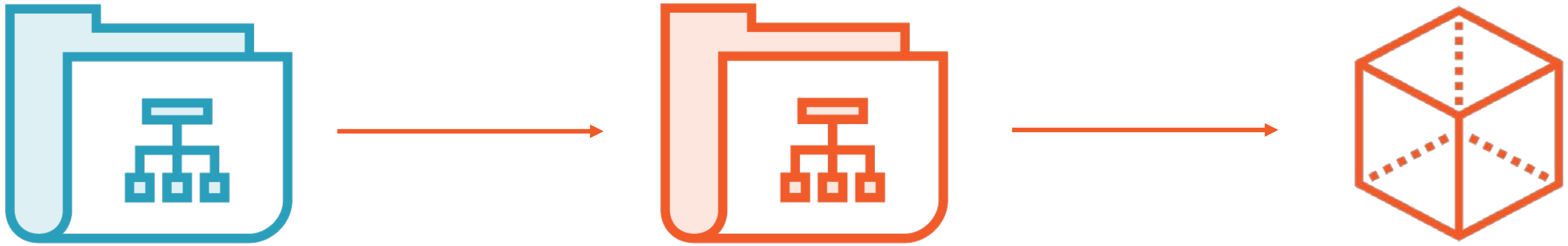
Investigate pom.xml

Identify project artifact type

Investigate the build node

Resources – Copying the JCR File System

Copying JCR_Root for Build



Project’s jcr_root

Target build folder

Generated package artifact

Maven Resources Plugin

```
<plugin>
   <artifactId>maven-resources-plugin</artifactId>
</plugin>
```

Maven Resources

```
<resources>
    <resource>
        <directory>
            ${basedir}/src/main/content/jcr_root
        </directory>
    </resource>
</resources>
```

Demo

Resources element

Check target location

Modify resources

Check target location

Content Package Maven Plugin

Content Package Plugin

Create packages for AEM

Upload/install packages to AEM

Using the Content Package Plugin

```
<groupId>com.day.jcr.vault</groupId>
<artifactId>content-package-maven-plugin</artifactId>
<version>x.x.xx</version>
<configuration>
    ...
</configuration>
```

Common Configurations

<targetURL>

<filters>

<group>

<embeds>

TargetURL Actions

Listing packages on remote

Build packages on remote

Uninstall and remove packages on remote

Demo

Investigate content-package plugin

Find our targetURL

AEM Project Profiles

Default Profiles from AEM Archetype

autoInstallPackage

autoInstallPackagePublish

autoInstallBundle

Package Delivery Profiles

Calls ‘install’ goal of the content-package-maven-plugin

Uploads and installs the created artifact

Executing the Install Goal

```
<executions>
    <execution>
        <id>install-package</id>
        <goals>
            <goal>install</goal>
        </goals>
    </execution>
</executions>
```

Bundle Delivery Profiles

Calls ‘install’ goal of the maven-sling-plugin

Uploads and installs the created artifact

Calling the Maven-Sling-Plugin

```
<groupId>org.apache.sling</groupId>
<artifactId>maven-sling-plugin</artifactId>
```

Configuring Maven Sling Plugin

```
<slingUrl>
http://${aem.host}:${aem.port}/crx/repository/crx.default
</slingUrl>


<slingUrlSuffix>
/apps/project-example/install/
</slingUrlSuffix>
```

Demo

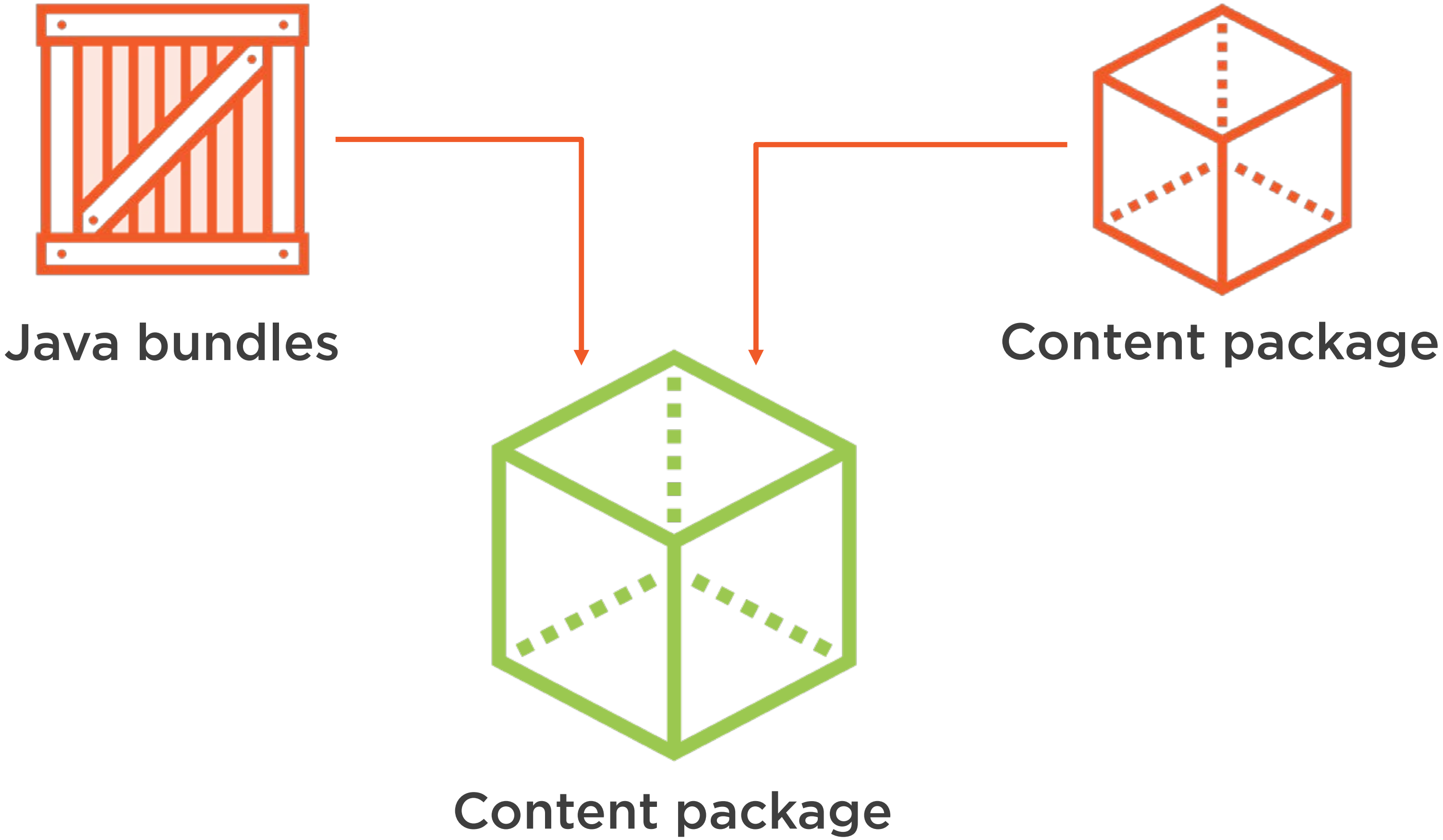
Investigate autoInstallPackage

Investigate autoInstallPackagePublish

Investigate autoInstallBundle

Embeds

Embeds in a Package



Using Embeds

<groupId> &
<artifactId>

Path/in/jcr/install

<dependency>

Using Embeds

```
<embed>
    <groupdId>embedGroup</groupdId>
    <artifactId>embedArtifact</artifactId>
    <target>apps/somewhere/install</target>
</embed>
```

Demo

Investigate embeds in content-package-maven-plugin

Locate the dependency block

Find the embed inside our package

Package Filters

Mapping Content with Filters

Contents in package

Contents in JCR

Adding Filters to a Package

<filterSource>

<filters>

Filters.xml

```
<filter root="/apps/myapp" mode="merge">
    <include pattern="/apps/myapp/install/" />
    <include pattern="/apps/myapp/components" />
    <exclude pattern="/apps/myapp/config/*" />
</filter>
```

Filter Node in Plugin

```
<filter>
    <root>/apps/myapp</root>
    <mode>merge</mode>
    <includes>
        <include>/apps/myapp/install/</include>
        <include>/apps/myapp/components</include>
    </includes>
    <excludes>
        <exclude>/apps/myapp/config/*</exclude>
    </excludes>
</filter>
```

Archetype generated projects are misleading as they handle filters through resources.

Demo

View the filters in our project

Change filter.xml

Point our plugin to the filter.xml

Add a filter to the plugin

View generated package

Package Filter Modes

Filter Mode

Strategy for handling content that already exists at that path in the JCR.

Available Filter Modes

Replace

Merge

Update

Replace Mode

Contents in the package mapped to the JCR root, replace any content in the JCR. This deletes JCR content that doesn't exist in the package!

Replace Mode Example

/apps/proj/
/apps/proj/compontent-1
/apps/proj/more-content
/apps/proj/compontent-3

Package

/apps/proj/
/apps/proj/compontent-1
/apps/proj/compontent-2
/apps/proj/more-content

Existing JCR

/apps/proj/
/apps/proj/compontent-1
/apps/proj/more-content
/apps/proj/compontent-3

JCR after install

⬛ Unmodified

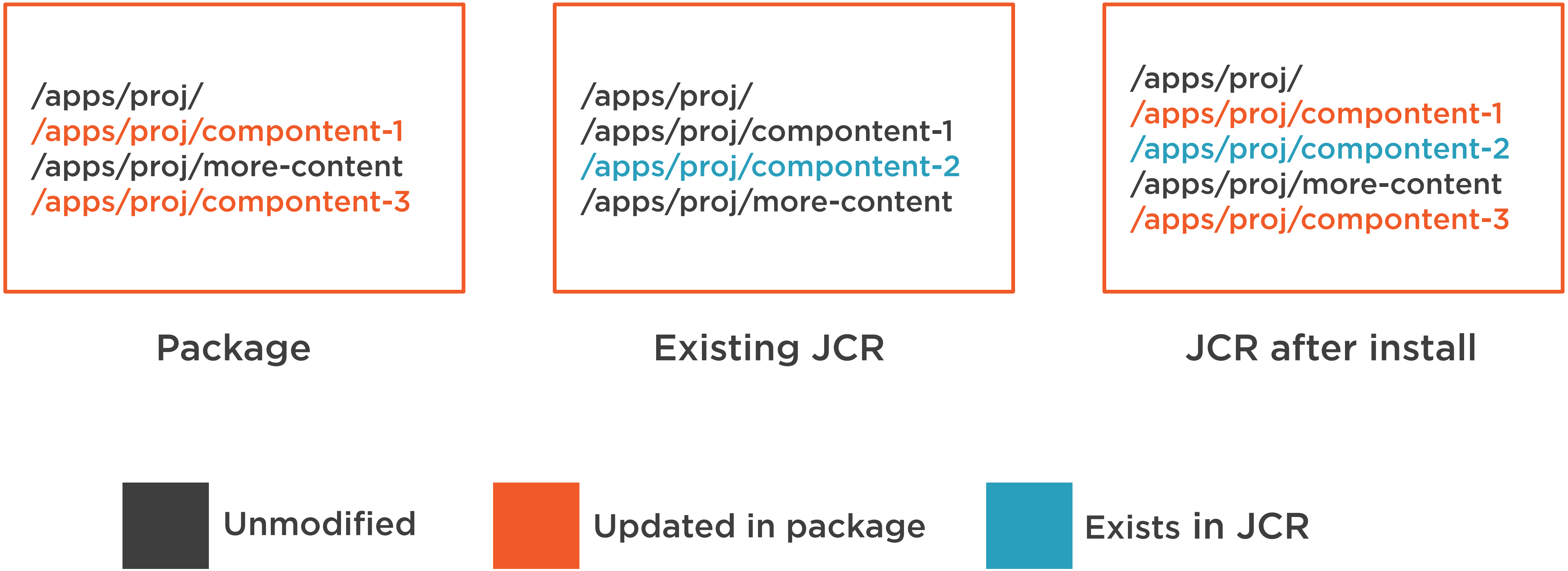
🟧 Updated in package

🟦 Exists in JCR

Update Mode

Contents in the package update existing content in the JCR. Nothing else. Nothing is deleted from the JCR.

Update Mode Example



/apps/proj/
/apps/proj/compontent-1
/apps/proj/more-content
/apps/proj/compontent-3

Package

/apps/proj/
/apps/proj/compontent-1
/apps/proj/compontent-2
/apps/proj/more-content

Existing JCR

/apps/proj/
/apps/proj/compontent-1
/apps/proj/compontent-2
/apps/proj/more-content
/apps/proj/compontent-3

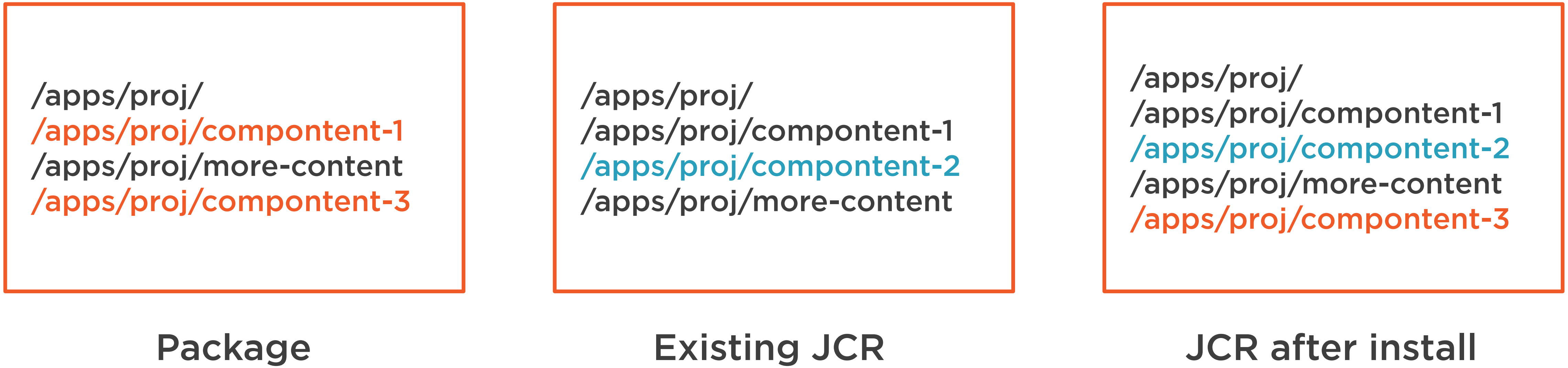
JCR after install

⬛ Unmodified 🟧 Updated in package 🟦 Exists in JCR

Merge Mode

Only content that doesn’t exist in the JCR already is added. Nothing is modified that currently exists in the JCR.

Merge Mode Example



/apps/proj/
/apps/proj/compontent-1
/apps/proj/more-content
/apps/proj/compontent-3

Package

/apps/proj/
/apps/proj/compontent-1
/apps/proj/compontent-2
/apps/proj/more-content

Existing JCR

/apps/proj/
/apps/proj/compontent-1
/apps/proj/compontent-2
/apps/proj/more-content
/apps/proj/compontent-3

JCR after install

⬛ Unmodified 🟧 Updated in package 🟦 Exists in JCR

Setting Mode in Filter.xml

```
<filter root=“/apps/myapp” mode=“merge” />
```

Setting Filter Mode in Plugin

```
<filter>
    <root>/apps/myapp</root>
    <mode>merge</mode>
</filter>
```

If no mode is selected, the *replace* mode will be used by default

Demo

Install simple project

Change the modes of our filter

Investigate the output during install

Summary

What’s really happening during a build

Content-package-maven-plugin

AEM specific profiles

Embedding artifacts into a package

Package filters

Filter modes