wcm.io Context-Aware Configuration

DATM-55

Technical Training – wcm.io

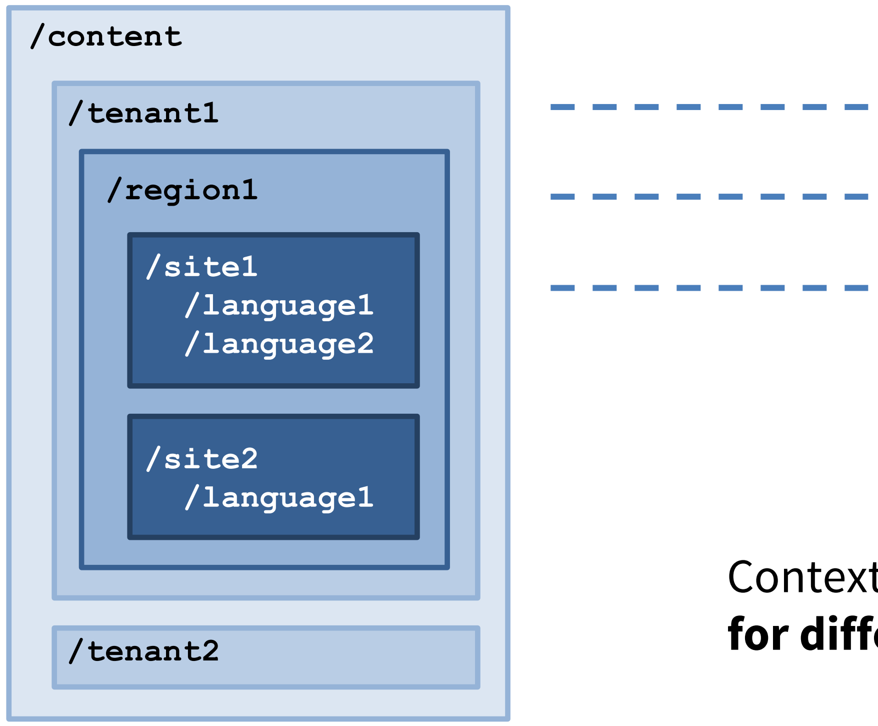
Last Updated: December 2021

https://training.wcm.io/caconfig/

What is Context-Aware Configuration

Short overview

Configuration example



Tenant-specific configuration

Region-specific configuration

Site-specific configuration

Context-aware = **different configuration for different subtrees in resource hierarchy**

Context-Aware Configuration

- Context-aware configurations are configurations that are **related to a content resource or a resource tree**, e.g. a web site or a tenant site.
- An application may need different configuration for different sites, regions and tenants = different contexts.
- Some parameters may be shared, so inheritance for nested contexts and from global fallback values is supported as well.

See also:

- Apache Sling documentation: Apache Sling Context-Aware Configuration
- diva-e Training: DATM-13 Sling Context-Aware Configuration

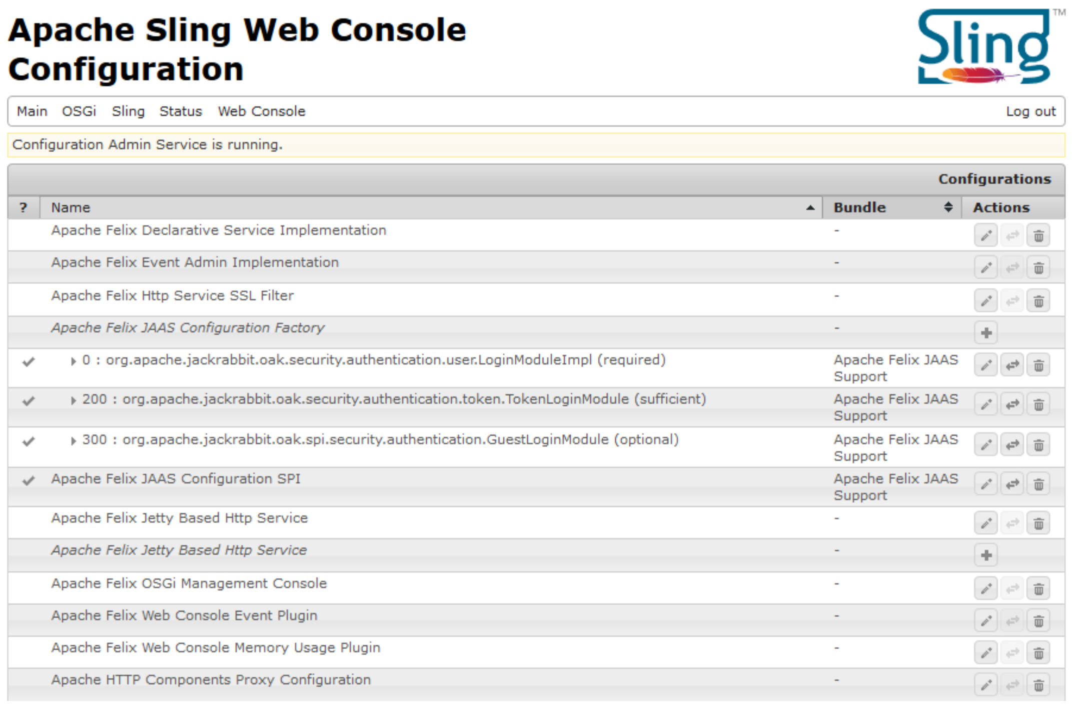
Configuration solutions in AEM

Configuration Solutions in AEM

Solution	Organization	Platform	System-level configuration	Context-aware configuration
OSGi configuration	OSGi	Sling, AEM	✓	✗
Cloud Service Configurations (CSC)	Adobe	AEM (since 5.5)	✗	✓
AEM ConfMgr	Adobe	AEM (since 6.1)	✗	✓
wcm.io Configuration 0.x	wcm.io	AEM (6.0 and up)	✗	✓
Apache Sling Context-Aware Configuration	Apache	Sling, AEM (6.1 and up)	✗	✓

- For system-level always OSGi is the standard solution
- For context-aware configuration different solutions emerged over the time

OSGi configuration



Apache Sling Web Console
Configuration

Configuration Admin Service is running.

Configurations

?	Name	Bundle	Actions
	Apache Felix Declarative Service Implementation	-	
	Apache Felix Event Admin Implementation	-	
	Apache Felix Http Service SSL Filter	-	
	Apache Felix JAAS Configuration Factory	-	
✓	0 : org.apache.jackrabbit.oak.security.authentication.user.LoginModuleImpl (required)	Apache Felix JAAS Support	
✓	200 : org.apache.jackrabbit.oak.security.authentication.token.TokenLoginModule (sufficient)	Apache Felix JAAS Support	
✓	300 : org.apache.jackrabbit.oak.spi.security.authentication.GuestLoginModule (optional)	Apache Felix JAAS Support	
✓	Apache Felix JAAS Configuration SPI	Apache Felix JAAS Support	
	Apache Felix Jetty Based Http Service	-	
	Apache Felix Jetty Based Http Service	-	
	Apache Felix OSGi Management Console	-	
	Apache Felix Web Console Event Plugin	-	
	Apache Felix Web Console Memory Usage Plugin	-	
	Apache HTTP Components Proxy Configuration	-	

- Editor GUI
- Flexible deployment: filesystem, repository, web console, factory configurations
- “Self-describing" with metadata
- Good API support (esp. in OSGi R6)
- Runmode-specific configuration

AEM ConfMgr

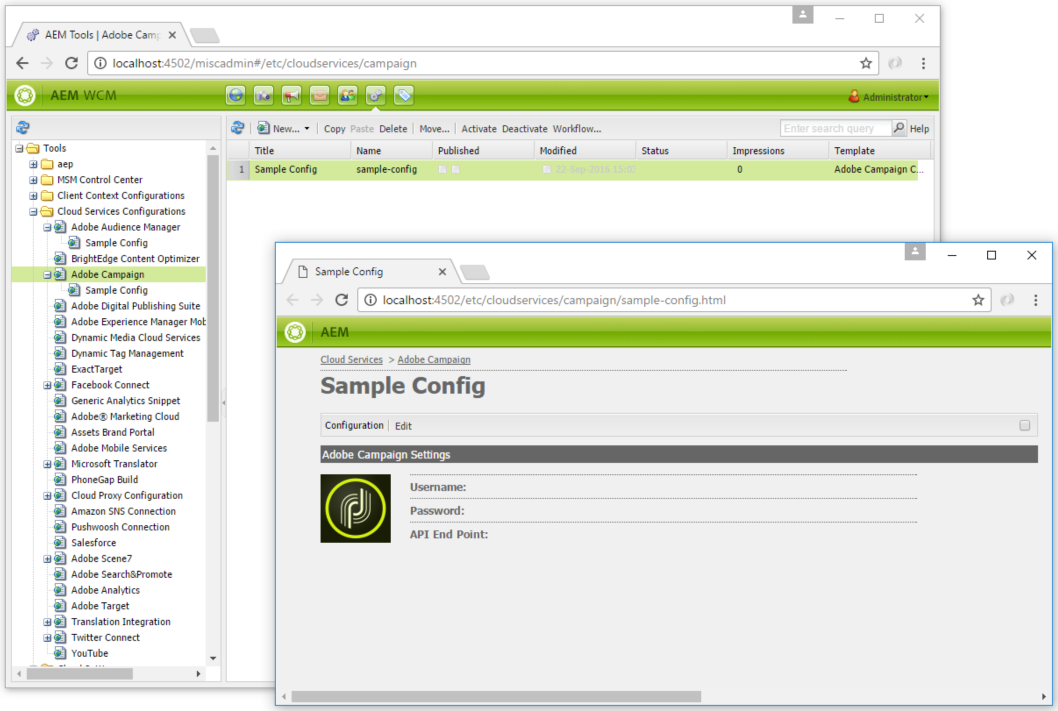
- Simple API
- Flexible inheritance support
- No Editor GUI
- Lacks documentation
- Used mainly by (some parts of) AEM itself
- Storage: `/conf`

- **Since AEM 6.3 replaced by Apache Sling Context-Aware Configuration**
  – AEM ConfMgr API still exists, but is deprecated and delegates to the Sling Context-Aware Configuration API internally

http://www.nateyolles.com/blog/2016/03/aem-slash-conf-and-confmgr

Cloud Service Configurations (CSC)

- Edit configuration via AEM templates
- Primary target: Adobe Marketing Cloud integrations
- Custom configurations possible as well
- Storage: `/etc/cloudservices`



- Initially created only to configure Adobe Marketing Cloud Solutions in AEM (hence the name)
- But can by used for application-specific purposes as well

https://experienceleague.adobe.com/docs/experience-manager-65/developing/extending-aem/extending-cloud-services/extending-cloud-config.html?lang=en

Configuration solution comparison

Feature	OSGi Config	AEM ConfMgr	AEM CSC	Sling CAConfig
Global / fallback configuration	✓	✓	✗	✓
Hierarchy-based inheritance	✗	✓	✗	✓
Property inheritance merging	✗	✗	✗	✓
Provide properties and data types	✓	✗	✓	✓
Additional metadata for editors	✓	✗	✓	✓
Define Configuration metadata via code	✓	✗	✗	✓
Key/value pairs (ValueMap)	✓	✓	✓	✓
Resource-based access	✗	✓	✓	✓
Map to Java class	✓	✗	✗	✓
Configuration collections	✓	✓	✗	✓
Editor GUI	✓	✗	✓	✓

Recommendation

- Use **OSGi configuration** for **system-level configuration**
- Use **Apache Sling Context-Aware Configuration** for the other configuration purposes
  - with the help of wcm.io Context-Aware Configuration Extensions and Editor
- Do no longer use AEM ConfMgr or wcm.io Configuration 0.x
- Use Cloud Service Configurations only for “Marketing-Cloud-like” integration use cases

Context-Aware Configuration in AEM

Sling Context-Aware Configuration in AEM

- AEM 6.3 is the first version that ships with Sling Context-Aware Configuration
  - But you should deploy the latest bundles
    https://wcm.io/caconfig/deploy-configure-caconfig-in-aem.html
  - Some additional OSGi configurations are required
- AEM 6.5 and AEMaaCS ship with the latest bundles

Out-of-the-box support since AEM 6.3

- Supports reading context-aware configuration:
  - Storage at **`/conf`**
  - Using the default content model from Sling Context-Aware Configuration
  - Using the content model from AEM ConfMgr
    (with configurations wrapped in **`cq:Page`** nodes)
- Supports writing context-aware configuration
  - Only using the default content model from Sling Context-Aware Configuration
- Implements some subtle additions to the resource inheritance logic to be backward-compatible with AEM ConfMgr
  - Lookup in all parent paths below **`/conf`**, even if not explicitly defined by a context configuration reference or context paths strategy
  - Special inheritance decider for **`mergeList`** property from AEM ConfMgr

Managing configuration in /conf

- All context-aware configuration is stored by default in **`/conf`**
- In AEM there is **no support in the GUI for editing or replicating** context-aware configuration
  - AEM 6.3 introduces a new tool “Configuration Browser”, but this allows only to create “structure” and not to manipulate the contained configuration. It is mainly target at template editor-related configuration, and does not have a “publish” button for replication.
  - The “Activate Tree” feature could be use for replication, but it is a bit tricky to use for context-aware configurations, and normally should not be accessible to anyone except the system administrator
- So, the only built-in support is:
  - Edit configurations in CRX DE Lite
  - Creating a package of **`/conf`** or a subtree of it and replicate it to the publisher

wcm.io Context-Aware Configuration

wcm.io Context-Aware Config Overview

wcm.io provides additional context-aware features:

- **Configuration Editor**
- **AEM-specific extensions** for context path strategies, persistence and overriding

Context-Aware Configuration Editor

wcm.io

Configuration Editor Features

- Manage Context-Aware Configuration by creating an editor page in the content context
- Manage singleton configuration, configuration collections and nested configurations
- Display all configuration metadata and default values
- Support all data types and arrays of values
- Control collection and property inheritance and support overridden values
- Allows to define custom widgets for configuration properties like pathbrowser
- It uses the Sling Context-Aware Management API internally

Placing configuration editor page

- The configuration editor is created as AEM page within the context, using the Configuration Editor template
- But it reads and writes the configuration from **`/conf`**
- When multiple contexts are nested an editor page is created for each of them

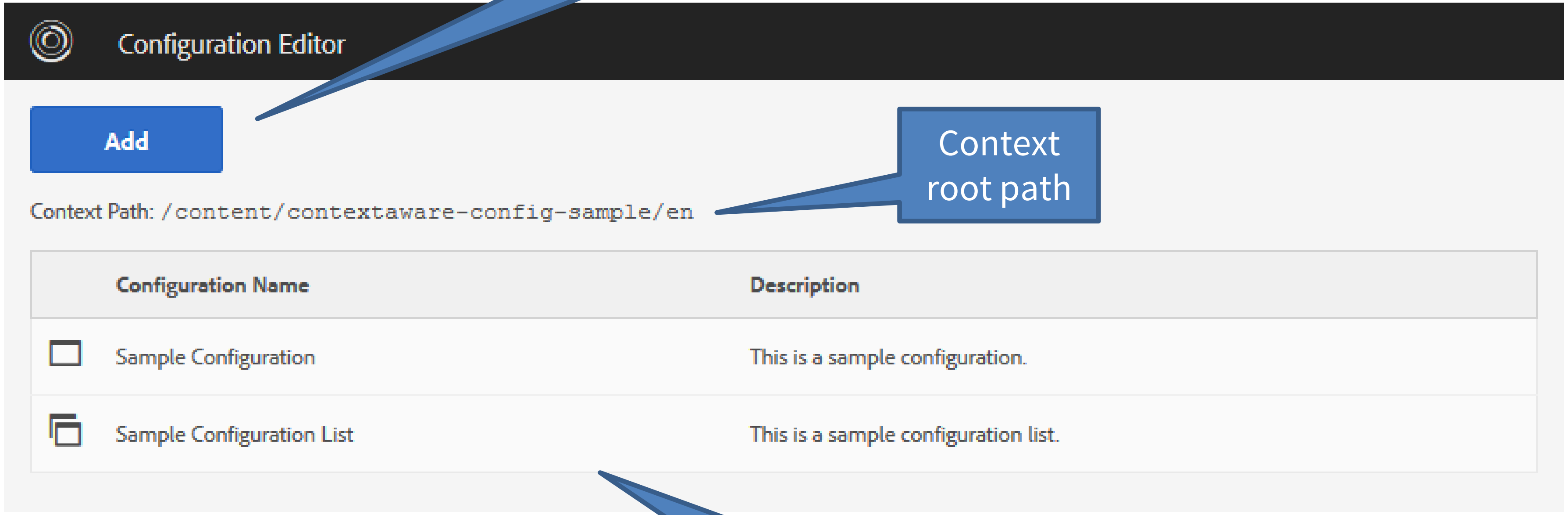
```
/content
    /mysite
      @sling:configRef = "/conf/mysite"
        /tools
           /config
```

Configuration editor page is created here
(anywhere within context subtree)

```
/conf
    /mysite
        /sling:configs
            /x.y.z.MyConfig
              @param1 = "value1"
```

Configuration is read from and written to /conf
(or whatever persistence strategy is configured)

Configuration overview



Context Path: /content/contextaware-config-sample/en

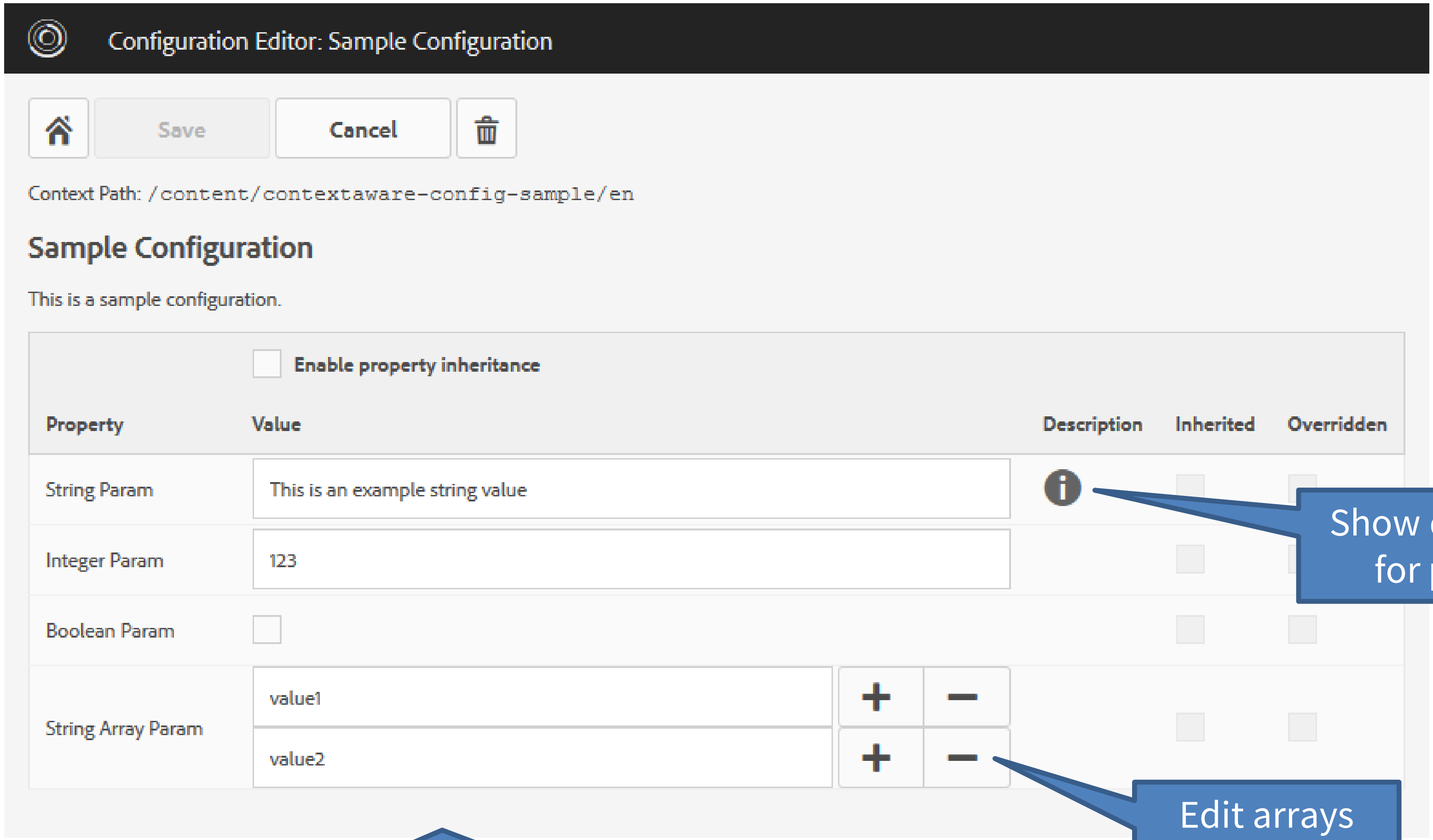
	Configuration Name	Description
☐	Sample Configuration	This is a sample configuration.
⧉	Sample Configuration List	This is a sample configuration list.

Enter data for configurations which do not yet exist

Context root path

Display configurations for which some configuration data already exists

Singleton configuration



Configuration Editor: Sample Configuration

Context Path: /content/contextaware-config-sample/en

Sample Configuration

This is a sample configuration.

☐ Enable property inheritance

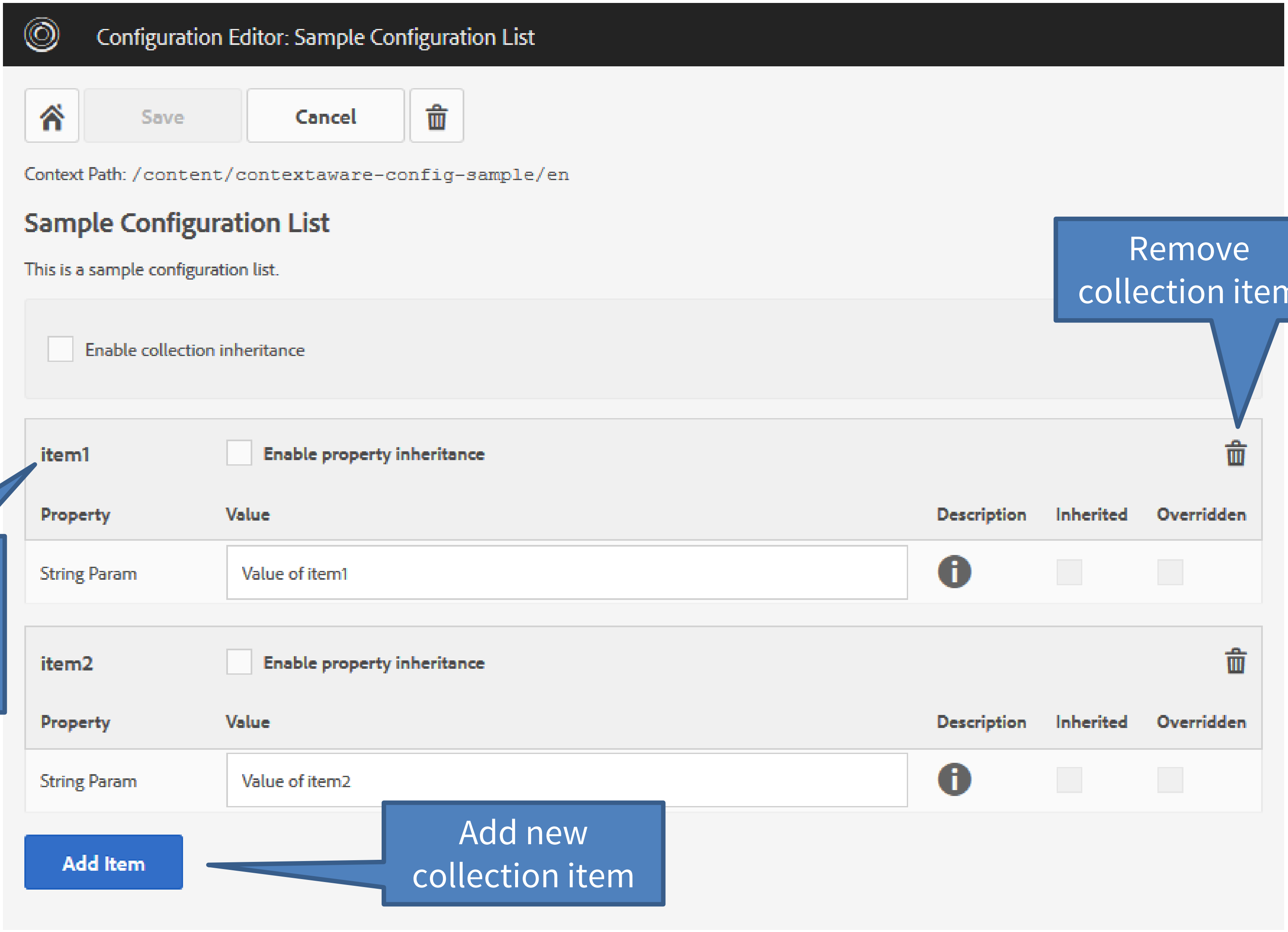
Property	Value	Description	Inherited	Overridden
String Param	This is an example string value	ⓘ	☐	☐
Integer Param	123		☐	☐
Boolean Param	☐		☐	☐
String Array Param	value1 / value2		☐	☐

Show description for property

Edit arrays of values

Displays all configuration properties with edit widgets matching it's data type.

Configuration collection


Configuration Editor: Sample Configuration List

Context Path: `/content/contextaware-config-sample/en`

Sample Configuration List

This is a sample configuration list.

☐ Enable collection inheritance

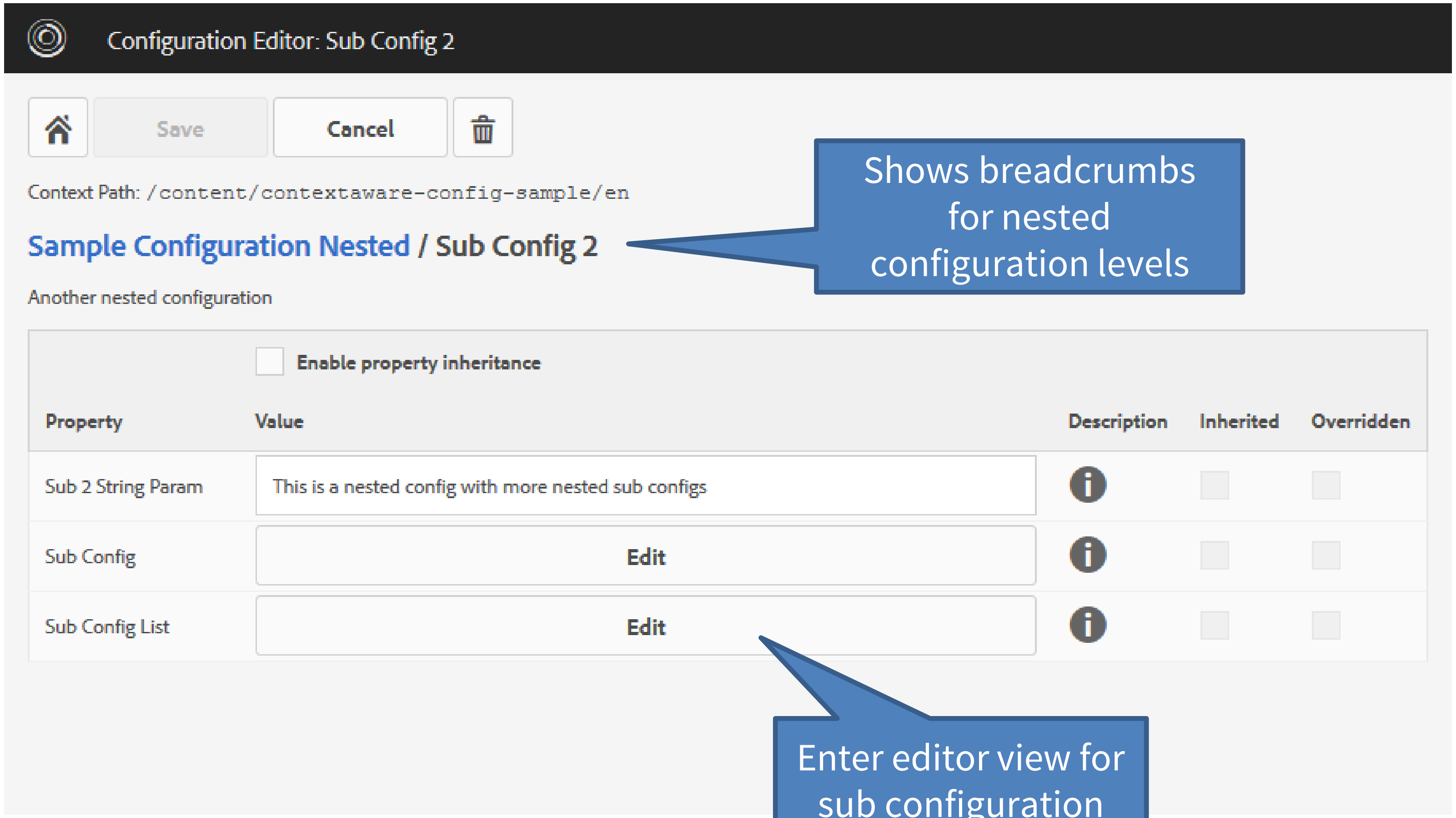
item1 ☐ Enable property inheritance

Property	Value	Description	Inherited	Overridden
String Param	Value of item1	●	☐	☐

item2 ☐ Enable property inheritance

Property	Value	Description	Inherited	Overridden
String Param	Value of item2	●	☐	☐

Nested configuration



Context Path: `/content/contextaware-config-sample/en`

Sample Configuration Nested / Sub Config 2

Another nested configuration

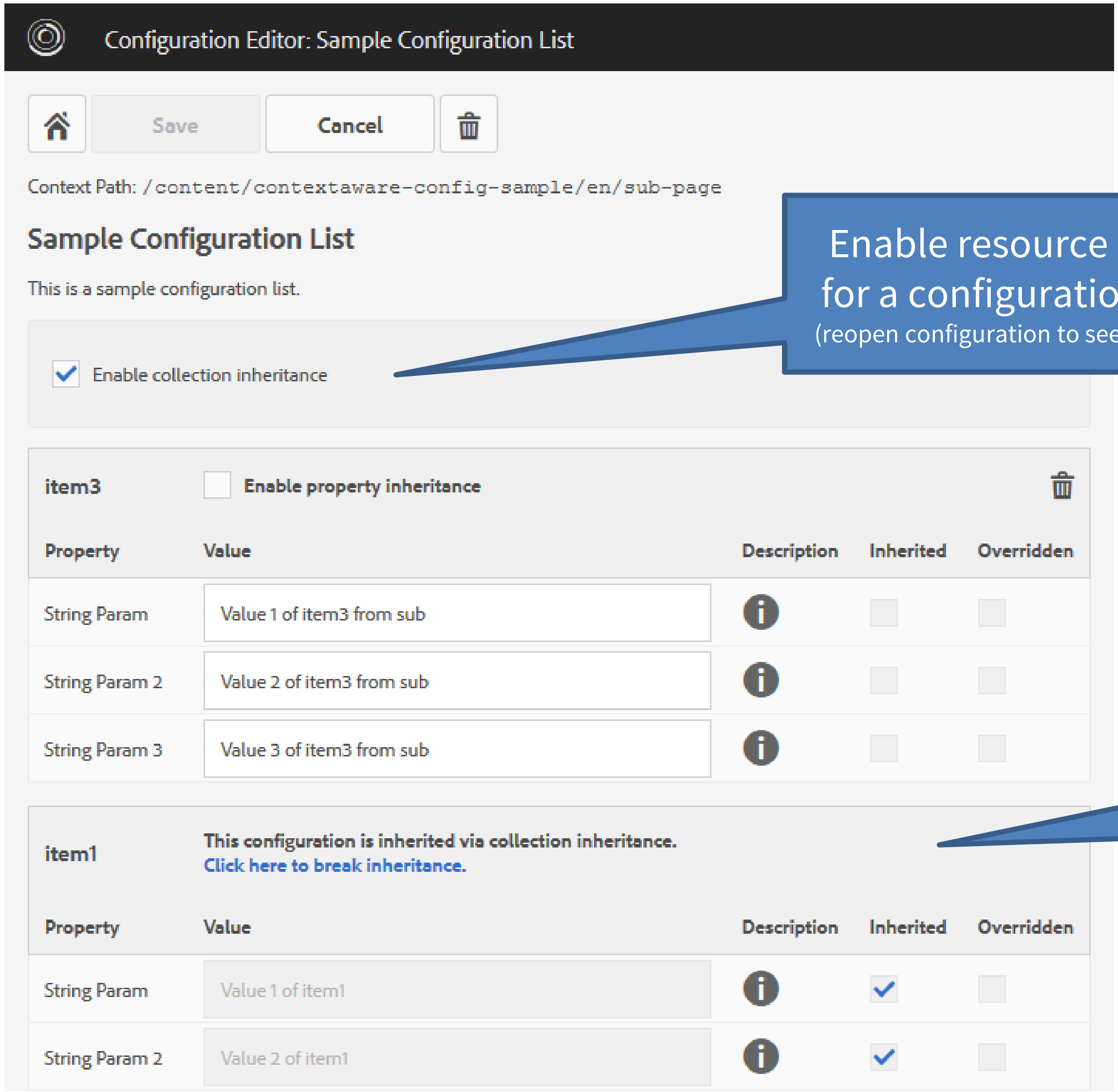
☐ Enable property inheritance

Property	Value	Description	Inherited	Overridden
Sub 2 String Param	This is a nested config with more nested sub configs	ⓘ	☐	☐
Sub Config	Edit	ⓘ	☐	☐
Sub Config List	Edit	ⓘ	☐	☐

Shows breadcrumbs for nested configuration levels

Enter editor view for sub configuration

Resource inheritance



Configuration Editor: Sample Configuration List

Save | Cancel

Context Path: /content/contextaware-config-sample/en/sub-page

Sample Configuration List

This is a sample configuration list.

[✓] Enable collection inheritance

item3 [ ] Enable property inheritance

Property	Value	Description	Inherited	Overridden
String Param	Value 1 of item3 from sub			
String Param 2	Value 2 of item3 from sub			
String Param 3	Value 3 of item3 from sub			

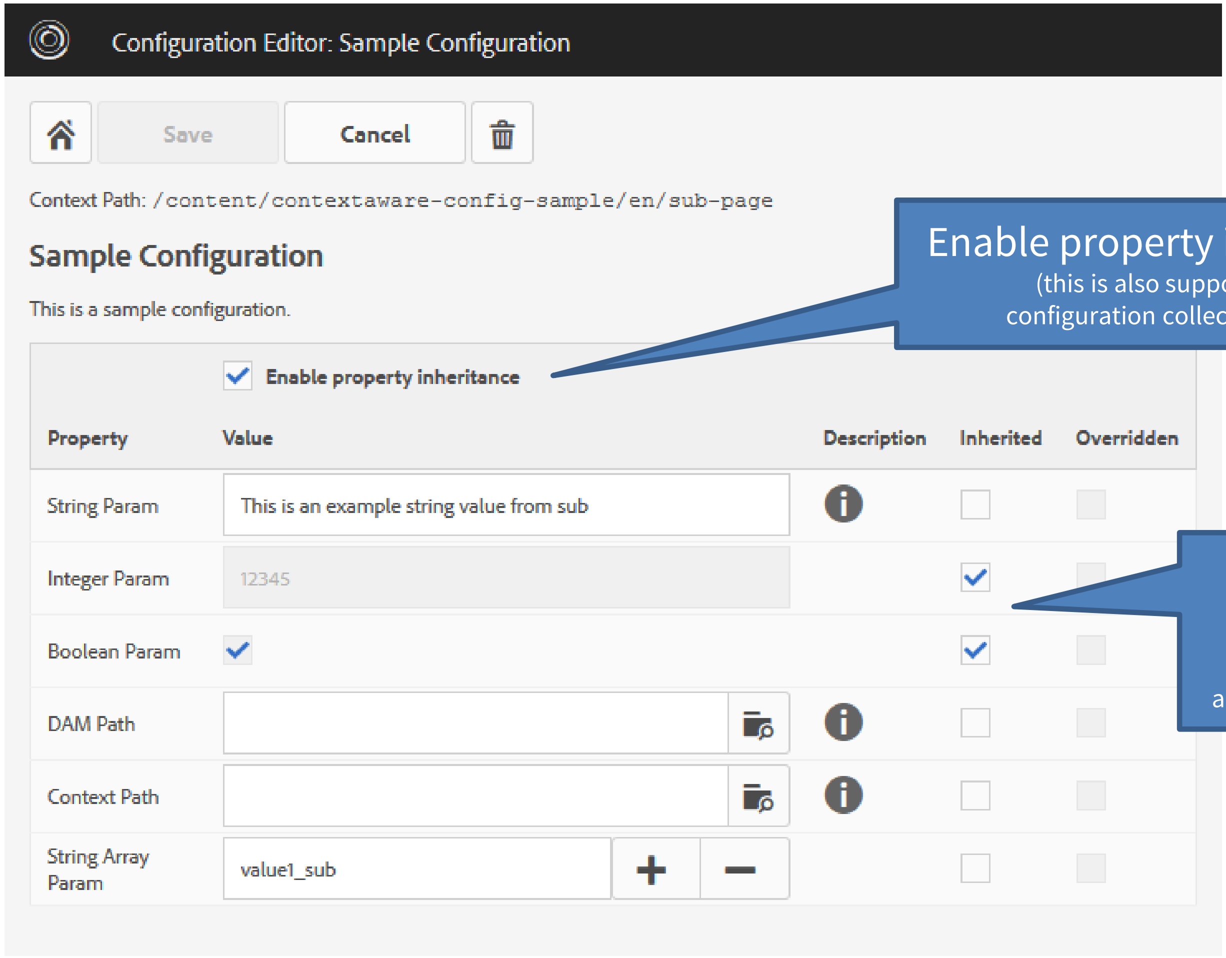
item1 This configuration is inherited via collection inheritance. Click here to break inheritance.

Property	Value	Description	Inherited	Overridden
String Param	Value 1 of item1		✓	
String Param 2	Value 2 of item1		✓	

Enable resource inheritance for a configuration collection
(reopen configuration to see inherited children)

This item is inherited.
(break inheritance to copy and edit it on this configuration level)

Property inheritance



Configuration Editor: Sample Configuration

Context Path: `/content/contextaware-config-sample/en/sub-page`

Sample Configuration

This is a sample configuration.

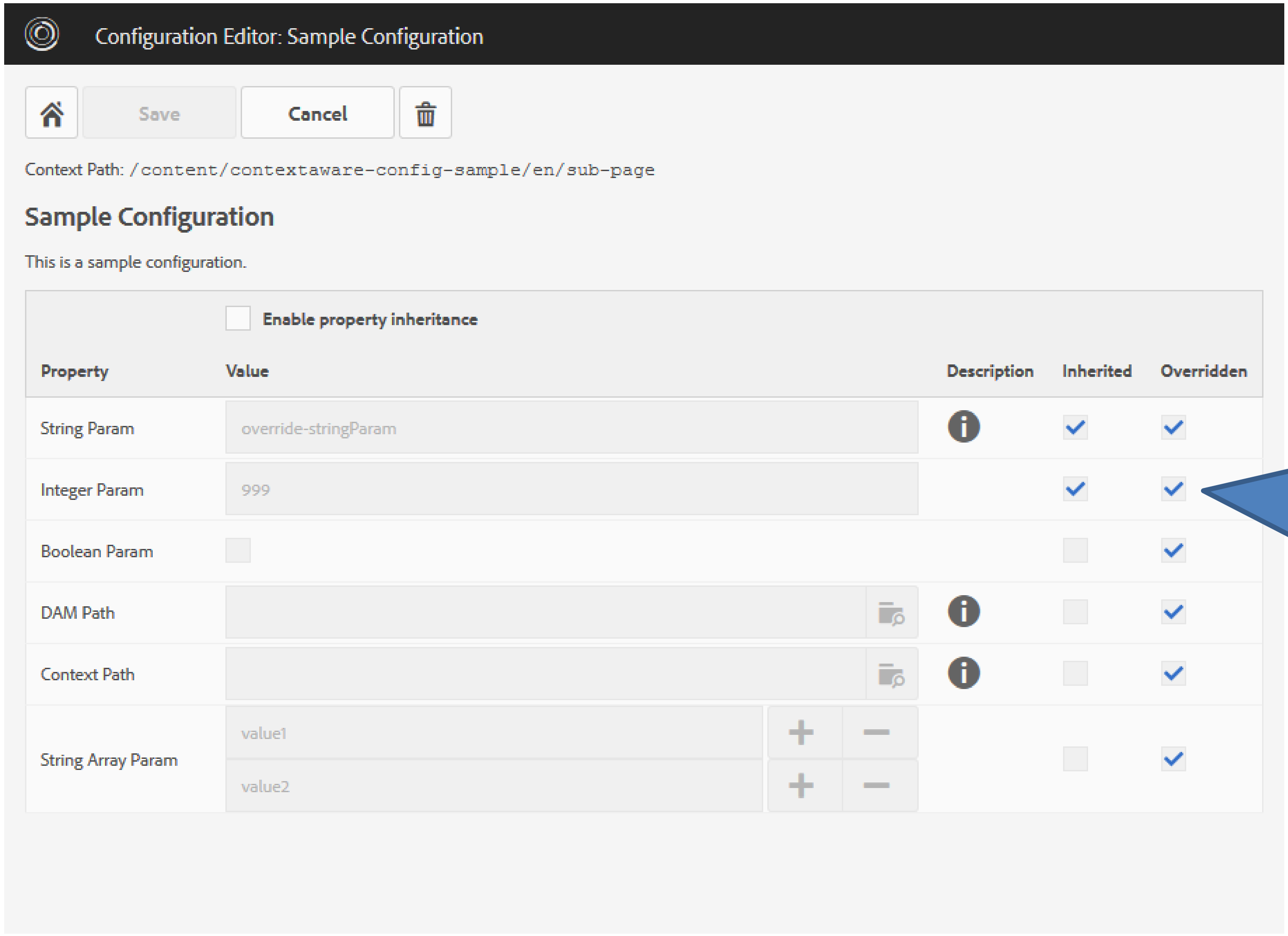
✓ Enable property inheritance

Property	Value	Description	Inherited	Overridden
String Param	This is an example string value from sub	●	☐	☐
Integer Param	12345		✓	☐
Boolean Param	✓		✓	☐
DAM Path		●	☐	☐
Context Path		●	☐	☐
String Array Param	value1_sub		☐	☐

Enable property inheritance
(this is also supported for configuration collection items)

Some properties are inherited.
(uncheck to overwrite with a new value on this configuration level)

Configuration override



Configuration Editor: Sample Configuration

Context Path: `/content/contextaware-config-sample/en/sub-page`

Sample Configuration

This is a sample configuration.

☐ Enable property inheritance

Property	Value	Description	Inherited	Overridden
String Param	override-stringParam	●	✓	✓
Integer Param	999		✓	✓
Boolean Param	☐		☐	✓
DAM Path		●	☐	✓
Context Path		●	☐	✓
String Array Param	value1 value2		☐	✓

When an override is configured for the current content path the properties are **read-only**.

Custom edit widgets

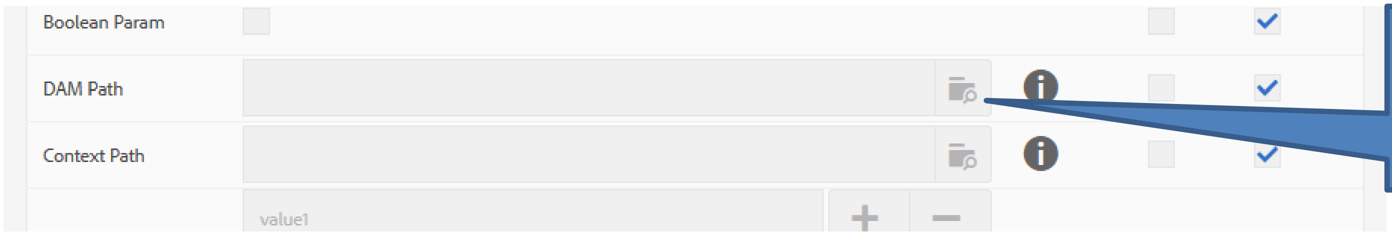
- You can define custom edit widgets for the configuration properties.
  - Currently only one “widgetType” is supported: “pathbrowser”

```
@Property(label = "DAM Path", property = {
     "widgetType=pathbrowser",
     "pathbrowserRootPath=/content/dam"
})
String damPath();

@Property(label = "Context Path", property = {
     "widgetType=pathbrowser",
     "pathbrowserRootPathContext=true"
})
String contextPath();
```

Use custom properties to configure the “widgetType” and it’s properties.

Sets root path to inner-most context-past



Boolean Param	
DAM Path	
Context Path	
	value1

Open path browser dialog

Integrate the editor into your application

- In most cases you will deploy the configuration editor bundle **`io.wcm.caconfig.editor`** together with your application.
- In this case you have to define your own template definition for it which controls where editor config pages can created – example:

```
{
  "jcr:primaryType": "cq:Template",
  "jcr:title": "My Application Configuration Editor",

  "allowedPaths": "^/content/myapp(/.*)?$",

  "jcr:content": {
    "jcr:primaryType": "cq:PageContent",
    "sling:resourceType": "/apps/wcm-io/caconfig/editor/components/page/editor"
  }
}
```

- Alternatively you can deploy an AEM package with a preconfigured template: **`io.wcm.caconfig.editor.package`**

Configuration editor sample application

If you want to try out the configuration editor on local AEM instance and test the different configuration use cases, you can use this sample application:

https://github.com/wcm-io/wcm-io-caconfig/tree/develop/sample-app

Use the script **`clean_install_deploy_package.sh`** to deploy the application and sample content to your AEM instances on port 4502.

Context-Aware Configuration Extensions

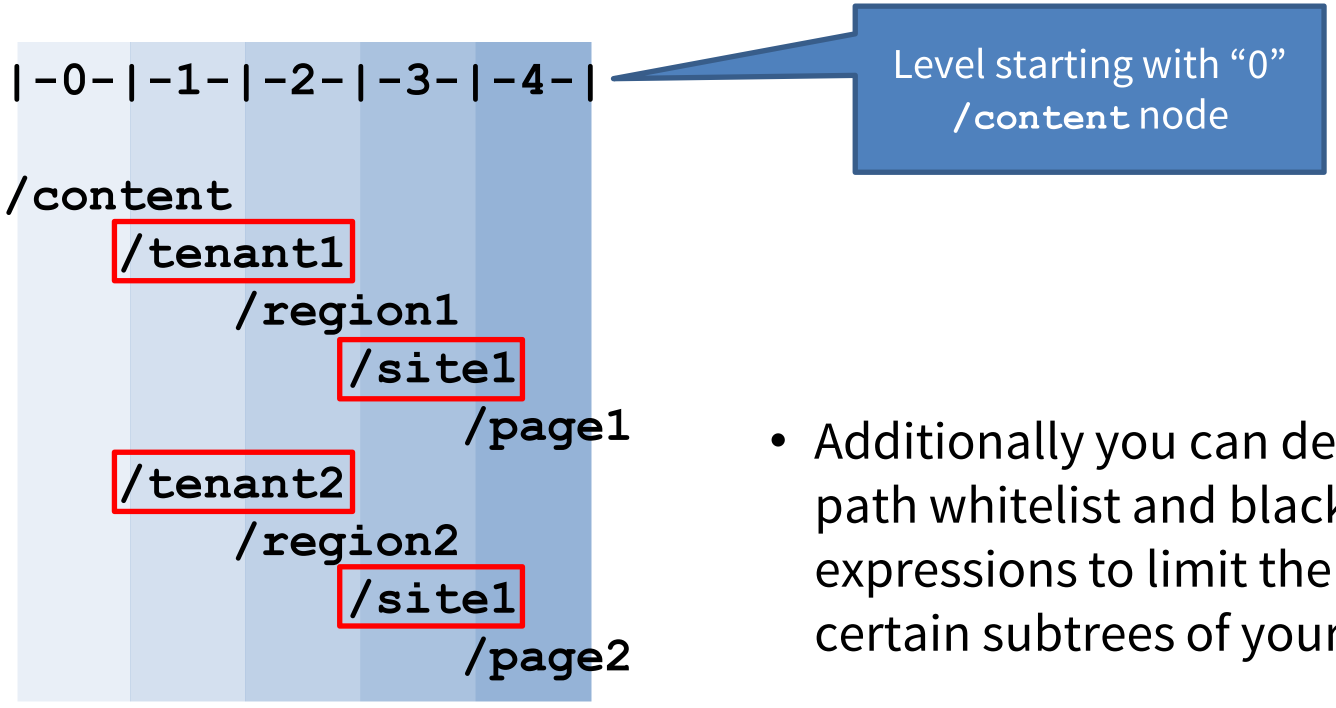
wcm.io

Context Path Strategies

- The Sling Context-Aware Configuration default implementation requires a **`sling:configRef`** property on the root of each context.
  - It's tedious and error-prone to define all those properties manually if you have a lot of sites
  - It does not enforce a well-ordered structure of site and configuration paths
- wcm.io provides alternative context path strategy implementations that detect the context roots automatically in a declarative way.
- You can have multiple strategies in place at the same time, separating them by path patterns or service ranking.

Context Path Strategy: Absolute Parents

- A fixed set of “absolute parent” path levels is used to define the context roots
- Example: Levels **1, 3** mark the following pages as context path roots



- Additionally you can define context path whitelist and blacklist regular expressions to limit the strategy to certain subtrees of your repository

Context Path Strategy: Root Templates

- Whenever a parent page uses a template matching a list of “root template paths” it defines the inner-most context root
- Example: Define the “Homepage Template”, min. level 1, max. level 4

```
|-0-|-1-|-2-|-3-|-4-|
/content
    /tenant1 <Structure Template>
        /region1 <Structure Template>
            /site1 <Homepage Template>
                /page1 <Content Template>
```

- All parent pages (or only those matching the templates) between min and max level up to a page with this configured template are detected as context paths.
- Additionally you can define context path whitelist expressions to limit the strategy to certain subtrees of your repository.

Context Path Strategies: Derive config paths

- Both “Absolute Parent” and “Root Template” context path strategies derive the configuration path from the context path.
- Regular expression groups and group references can be used for this

Example:

contextPathRegex	= `"^/content(/.+)$"`
configPathPatterns	= `["/conf$1"]`
Context root path	= `/content/tenant1/region1/site1`
Derived configuration path	= `/conf/tenant1/region1/site1`

- You can define multiple configPathPatterns – the paths are used from last to first for reading configuration, only the last one for writing.

Persistence Strategies

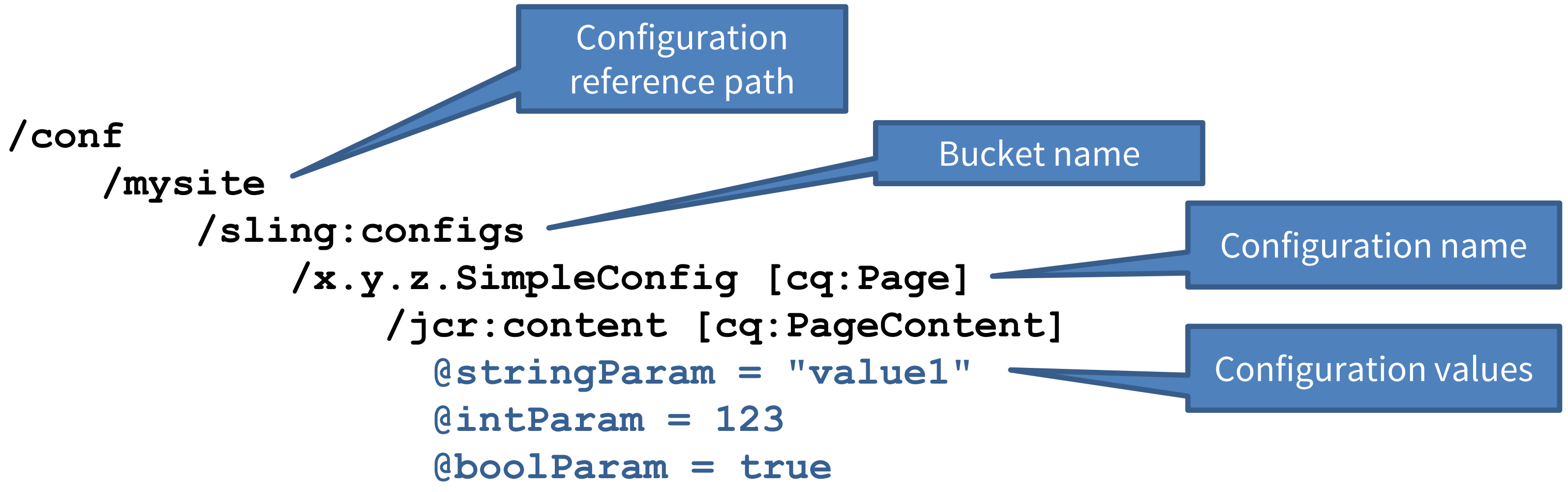
- By default Sling Context-Aware Configuration stores configuration in a hierarchy of nodes below `/conf` using `nt:unstructured` node types. This is simple enough, but it makes it difficult to apply operations like replication on it in AEM.
- Thus it would be good when configuration can be stored in cq:Page nodes as it is done by the “AEM ConfMgr” for AEM. AEM ships with such an Persistence Strategy, but it only supports read access to configuration, no write access.
- wcm.io provides additional persistence strategy implementations.

Persistence Strategy: AEM Page

- Stores configurations in **`cq:Page/jcr:content`** nodes instead of **`nt:unstructured`**
- Makes it easier to replicate them to publish individually
- Uses similar content model as AEM ConfMgr
- Disabled by default, can be enabled via OSGi configuration

Persistence Strategy: AEM Page

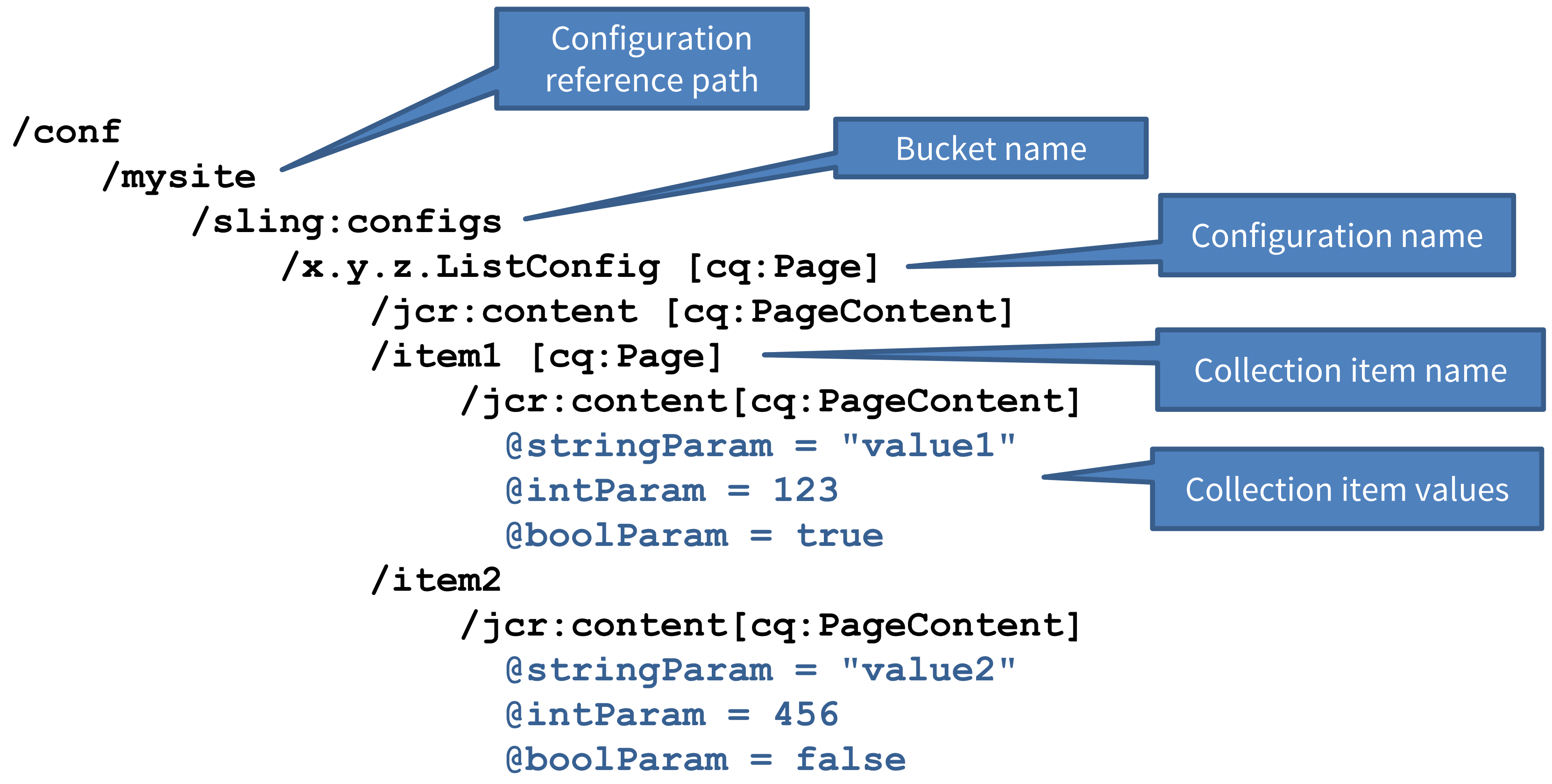
Example resource structure for a **singleton configuration**:



```
/conf
    /mysite
        /sling:configs
            /x.y.z.SimpleConfig [cq:Page]
                /jcr:content [cq:PageContent]
                  @stringParam = "value1"
                  @intParam = 123
                  @boolParam = true
```

Persistence Strategy: AEM Page

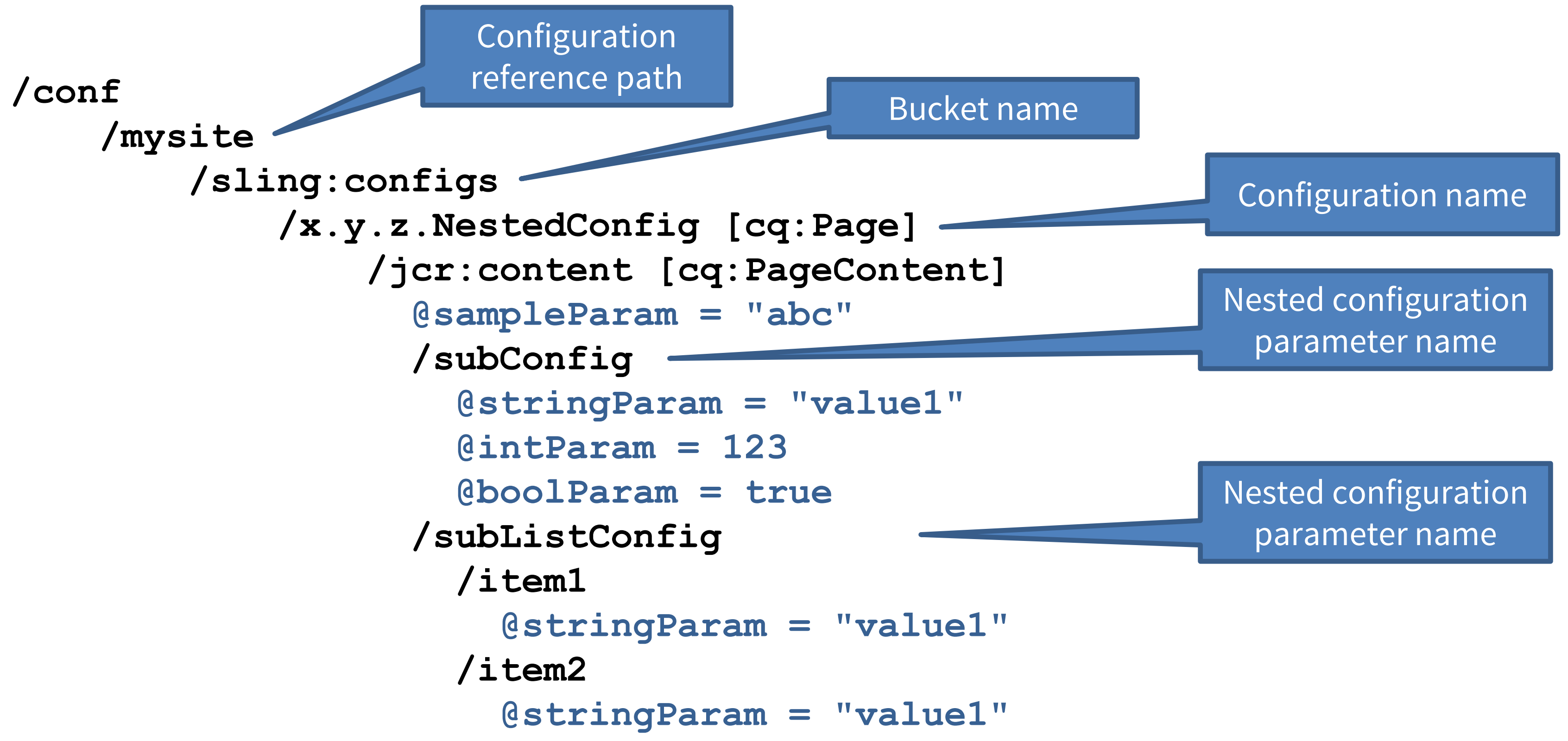
Example resource structure for a **configuration collection**:



```
/conf
    /mysite
        /sling:configs
            /x.y.z.ListConfig [cq:Page]
                /jcr:content [cq:PageContent]
                /item1 [cq:Page]
                    /jcr:content[cq:PageContent]
                      @stringParam = "value1"
                      @intParam = 123
                      @boolParam = true
                /item2
                    /jcr:content[cq:PageContent]
                      @stringParam = "value2"
                      @intParam = 456
                      @boolParam = false
```

Persistence Strategy: AEM Page

Example resource structure for a **nested configuration**:


Configuration reference path
Bucket name
Configuration name
Nested configuration parameter name
Nested configuration parameter name

```
/conf
    /mysite
        /sling:configs
            /x.y.z.NestedConfig [cq:Page]
                /jcr:content [cq:PageContent]
                  @sampleParam = "abc"
                  /subConfig
                    @stringParam = "value1"
                    @intParam = 123
                    @boolParam = true
                  /subListConfig
                    /item1
                      @stringParam = "value1"
                    /item2
                      @stringParam = "value1"
```

Persistence Strategy: Tools Config Page

- Stores configurations in `tools/config` pages as **part of the content**, and **not** below `/conf`
- Advantages:
  - Configuration can be packaged or replicated easily together with content
  - Configuration can be activated, versioned etc. directly from Author GUI
  - Same concept as in wcm.io Configuration 0.x
- Disadvantages:
  - Configuration cannot be easily protected via ACLs
  - Not following best-practices (mixes content and configuration)
- Disabled by default, can be enabled via OSGi configuration
- For detailed setup instructions see wcm.io documentation

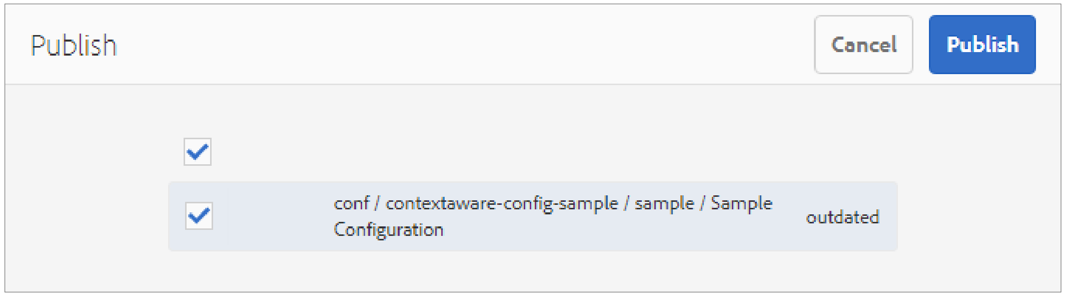
Override Provider: Request Header

- Injects configuration overrides from HTTP headers incoming HTTP requests.
- This is useful on QA instances with automated tests which expect a certain context-aware configuration.
  - **It should never be activated on production instances.**
- Via the “Header Name” configuration property the name of the header is defined. The header can be included multiple times in the request, each containing an configuration override string.
- This provider is deactivated by default.

Reference Provider

- The ReferenceProvider is an AEM service interface to report reference to AEM pages (e.g. AEM assets referenced by a page). wcm.io CAConfig Configuration Extensions provides an implementation for configuration pages below /conf.

- If you use the "AEM Page" persistence strategy the configuration is stored as AEM pages below /conf. If they are outdated they are offered for publication when you activate a page of a related configuration context:



conf / contextaware-config-sample / sample / Sample Configuration outdated

- Enabled by default, can be disabled by configuration.

Unit Test Support

Unit Tests with Context-Aware Configuration

- When your code depends on wcm.io Context-Aware Configuration Extensions and you want to write **AEM Mocks**-based unit tests running against the Context-Aware configuration implementation you have to register the proper OSGi services to use them.

- To make this easier, a
  “**wcm.io Context-Aware Configuration Mock Helper**“
  is provided which does this job for you.

```
<dependency>
  <groupId>io.wcm</groupId>
  <artifactId>io.wcm.testing.wcm-io-mock.caconfig</artifactId>
  <scope>test</scope>
</dependency>
<dependency>
  <groupId>org.apache.sling</groupId>
  <artifactId>org.apache.sling.testing.caconfig-mock-plugin</artifactId>
  <scope>test</scope>
</dependency>
```

You need both plugins – from Sling and wcm.io.

Unit test example

```
import static io.wcm.testing.mock.wcmio.caconfig.ContextPlugins.WCMIO_CACONFIG;
import static org.apache.sling.testing.mock.caconfig.ContextPlugins.CACONFIG;

public class MyTest {

  @Rule
  public AemContext context = new AemContextBuilder()
      .plugin(CACONFIG)
      .plugin(WCMIO_CACONFIG)
      .build();

  @Before
  public void setUp() {
    // register configuration annotation class
    MockContextAwareConfig.registerAnnotationPackages(context, "com.myapp.config");

    // shortcut for registering a context path strategy for unit test
    MockCAConfig.contextPathStrategyRootTemplate(context, "/apps/myapp/templates/home");
  }

  ...

}
```

This plugs in the necessary Context-Aware configuration setup/teardown methods.

Helper method for quickly setting up a context path strategy.

Recommendations for AEM projects

Recommendations for AEM projects

- Use wcm.io Context-Aware Configuration Editor
  - Otherwise, you can edit the configuration only via CRX DE Lite
  - Define your own template definition to control where it can be created
  - Disable it on publish via OSGi configuration
- Use wcm.io Context-Aware Configuration Extensions
  - Use “Root Template” or “Absolute Parent” context path strategy
  - Use “AEM Page” persistence strategy
- Apply metadata (labels, descriptions) to your configuration classes
  - It’s helpful for the user when using the configuration editor

ACLs

By default, most users have no read access to `/conf`. When you store context-aware configurations in this folder you need to setup proper ACLs on author and publish side.

- Be as explicit as possible and grant ACLs only the required subtrees of `/conf`, and only to the required groups
- On the author side:
  - all author users should have **jcr:read** access to subtree.
    Users allowed to change and publish configurations need:
    jcr:versionManagement, crx:replicate, rep:write, jcr:lockManagement
  - Access rights for `version-manager-service`:
    jcr:versionManagement, rep:write
- On the publish side the `everyone` user needs **jcr:read** access.