

adaptTo()

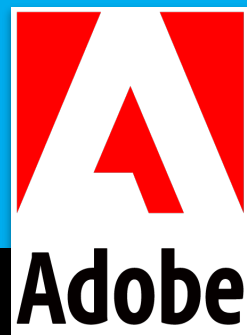
EUROPE'S LEADING AEM DEVELOPER CONFERENCE

27th – 29th SEPTEMBER 2021

repoint

a mini-language for content repository initialization

Bertrand Delacrétaz - Adobe & Apache - @bdelacretaz - grep.codeconsult.ch



Images: stock.adobe.com, unless otherwise specified - slides revision: 2021-09-29



what's repointit?

how to use it?

how to write a parser?

test-driven software?



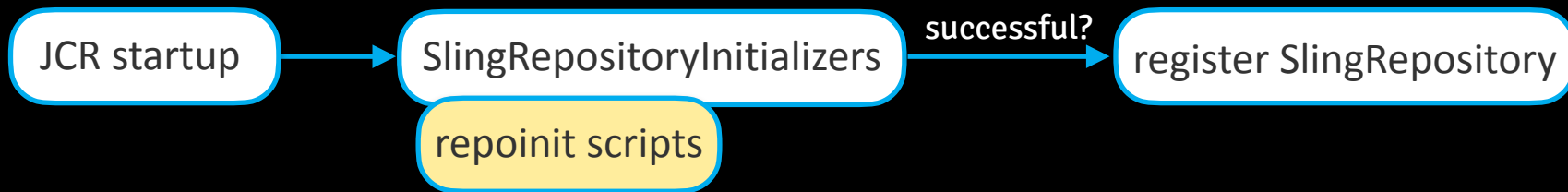
A close-up shot of a person's lower body. They are wearing bright pink pants and black rubber boots. They are walking through a muddy, leaf-covered ground, and their boots are splashing mud.

what is repointit?



repointit?

An Apache Sling mini-language for content repository initialization



```
set ACL on /libs,/apps
  allow jcr:read for alice, bob
  allow jcr:all for content-admins
end
```

<https://sling.apache.org/documentation/bundles/repository-initialization.html>

Quick, set it up before they see it!

SlingRepositoryInitializer

The `SlingRepositoryInitializer` is a very simple service interface, available from version 2.4.0 of the `org.apache.sling.jcr.api` and `org.apache.sling.jcr.base` bundles.

```
public interface SlingRepositoryInitializer {  
    public void processRepository(SlingRepository repo) throws Exception;  
}
```

Services that implement this interface are called when setting up the JCR-based `SlingRepository` service, before registering it as an OSGi service.

They are called in increasing order of their `service.ranking` service property, which needs to be an `Integer` as usual.

If any of them throws an Exception, the `SlingRepository` service is not registered.

<https://sling.apache.org/documentation/bundles/repository-initialization.html>

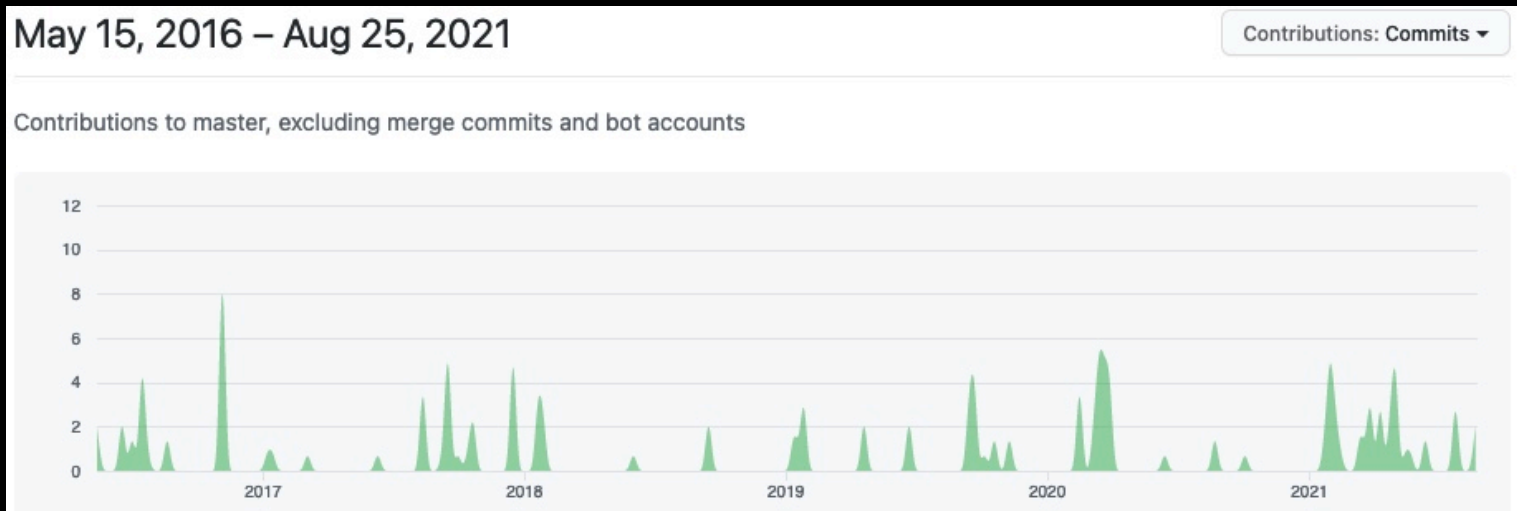
A close-up shot of a person's lower body as they walk through a muddy, leaf-strewn path. They are wearing bright pink pants and black rubber boots. The ground is covered in brown leaves and thick mud.

history

A close-up shot of a person's lower body as they walk through a muddy, leaf-strewn path. They are wearing black rubber boots. The ground is covered in brown leaves and thick mud, and the person is splashing through it.

history

"We need to initialize a number of things when starting Sling on a new or existing content repository"
January 2016



<https://issues.apache.org/jira/browse/SLING-5449>

(great) discussion: <https://s.apache.org/repointit-2016>

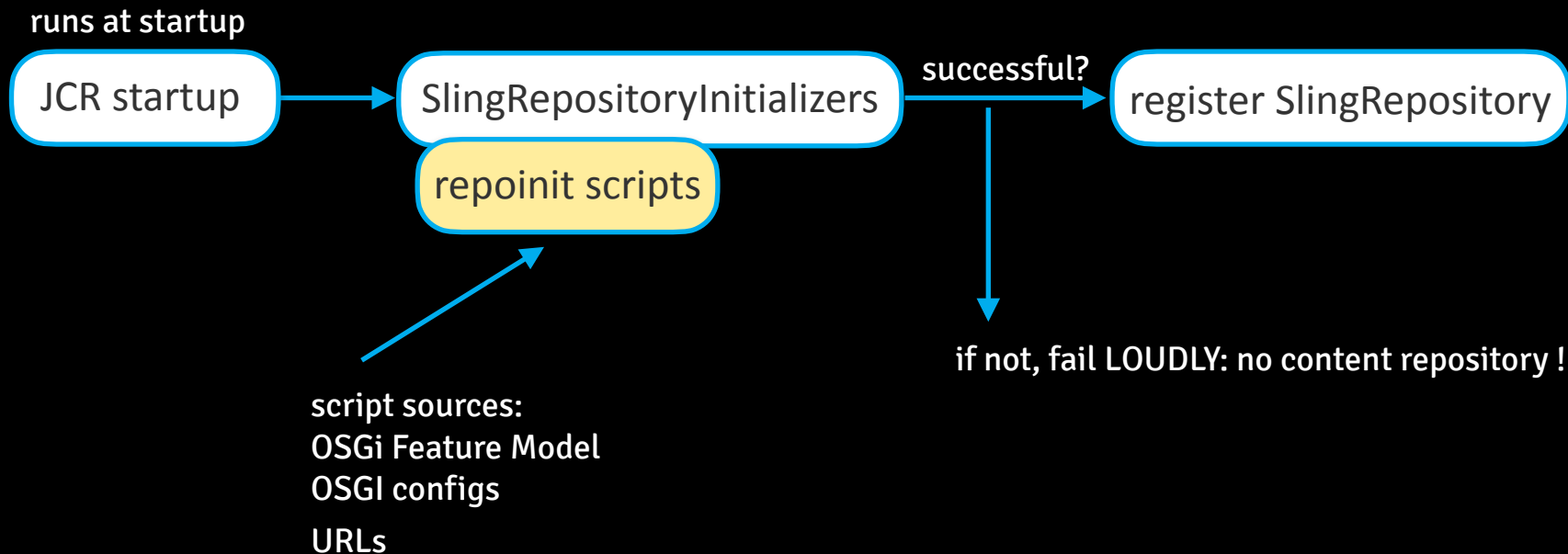
syntax idea: <https://issues.apache.org/jira/browse/JCRVLT-61>- credits to Tobias Bocanegra

A close-up shot of a person's lower body and legs. They are wearing bright pink pants and black rubber boots. They are walking through a muddy, leaf-covered ground, and their boots are splashing mud.

usage

A close-up shot of a person's lower body and legs. They are wearing black rubber boots and are walking through a muddy, leaf-covered ground. Their boots are splashing mud, and the ground is covered in brown leaves and puddles.

usage



also: explicit parsing and execution

```
public class RepointExampleExecutor {  
  
    private Session jcrSession;  
  
    @Inject  
    private RepoInitParser parser;  
  
    @Inject  
    private JcrRepoInitOpsProcessor processor;  
  
    public void parseAndExecute(String scriptName) throws Exception {  
        try(InputStream is = getClass().getResourceAsStream(scriptName)) {  
            List<Operation> ops = parser.parse(new InputStreamReader(is, "UTF-8"));  
            processor.apply(jcrSession, ops);  
            jcrSession.save();  
        }  
    }  
}
```

If needed for
specific
use cases..

A close-up shot of a person's lower body as they walk through a muddy, leaf-strewn path. They are wearing bright pink pants and black rubber boots. The ground is covered in brown leaves and thick mud.

examples

A close-up shot of a person's legs in black rubber boots, splashing through a deep, muddy puddle. The mud is thick and brown, and the person's legs are partially submerged. The background is a muddy path covered with brown leaves.

PACKED

examples section ahead
feel free to ask questions!

RAISE
YOUR
HAND



la doc! la doc!



<https://sling.apache.org/documentation/bundles/repository-initialization.html>



[Home](#) / [Documentation](#) / [Bundles](#)

[repoint](#)[jcr](#)[repository](#)

Repository Initialization (repoint)

DOCUMENTATION

- Overview
- Getting Started
- The Sling Engine
- Development
- Bundles
- Tutorials & How-Tos
- Maven Plugins
- Configuration

API DOCS

- Sling 11
- Sling 10
- Sling 9
- All versions

SUPPORT

- Wiki
- FAQ
- Sitemap

PROJECT INFO

- Downloads
- License
- News
- Releases
- Issue Tracker
- Links
- Contributing
- Project Information
- Security

SOURCE

- Repositories
- Git at Apache

APACHE SOFTWARE FOUNDATION

- Thanks!
- Become a Sponsor
- Buy Stuff



September 21-23
www.apachecon.com



TABLE OF CONTENTS

- SlingRepositoryInitializer
- The 'repoint' Repository Initialization Language
 - Notes on Repository Initializer Config Files
- Providing repoint statements from the Sling provisioning model or other URLs
 - References to Sling Provisioning Model additional sections
 - References to URLs providing raw repoint statements
- Providing repoint statements from OSGi factory configurations
- Appendix
 - Appendix A: repoint syntax: parser test scenarios
 - Repoint parser test scenarios

The `SlingRepositoryInitializer` mechanism allows for running code before the `SlingRepository` service is registered.

This is useful for initialization and content migration purposes.

Please be aware of potential clustering and coordination issues when using this mechanism, if your environment lets several Sling instances access the same content repository you'll need to implement a synchronization mechanism for such operations.

SlingRepositoryInitializer

The `SlingRepositoryInitializer` is a very simple service interface, available from version 2.4.0 of the `org.apache.sling.jcr.api` and `org.apache.sling.jcr.base` bundles.

```
public interface SlingRepositoryInitializer {  
    public void processRepository(SlingRepository repo) throws Exception;  
}
```

Services that implement this interface are called when setting up the JCR-based `SlingRepository` service, before registering it as an OSGi service.

They are called in increasing order of their `service.ranking` service property, which needs to be an `Integer` as usual.

If any of them throws an Exception, the `SlingRepository` service is not registered.

The 'repoint' Repository Initialization Language

The `org.apache.sling.repoint.parser` implements a mini-language meant to create paths, service users and Access Control Lists in a content repository, as well as registering JCR namespaces and node types.



Apache Sling repoint - Bertrand Delacrétaiz, Adobe - September 2021

14

create paths and set properties

```
create path /one/two/three
```

```
create path (sling:Folder) /var/disc(nt:unstructured)/afolder
```

```
create path /five(mixin nt:art)/step(mixin nt:dance)/two/steps
```

```
set properties on /pathA, /path/B
```

```
  set sling:ResourceType{String} to /x/y/z
```

```
  default someLong{Long} to 42
```

```
  set someFlag{Boolean} to true
```

```
  default someDate{Date} to "2020-03-19T11:39:33.437+05:30"
```

```
  set customMultiValueStringProp to test1, test2
```

```
end
```

manage (service) users

```
create service user bob,alice, tom21
```

```
create service user leo with path /some/absolute/path
```

```
delete service user Leonardo,Winston_32
```

```
disable service user svcA : "This explains why it's disabled"
```

```
create user AF with forced path /path/user/AF
```

```
create user userC with password plain_text_only_for_testing
```

```
delete user userC
```


groups

```
create group groupa  
create group groupb with path /thePathF  
create group groupc with forced path /thePathG
```

```
add user1,user2 to group groupD
```

```
remove user3,user5 from group groupE  
delete group groupa
```

```
set properties on authorizable(groupD)  
  set stringProp to "hello, group D"  
end  
set properties on authorizable(user3)/nested  
  set stringProp to "hello, nested!"  
end
```

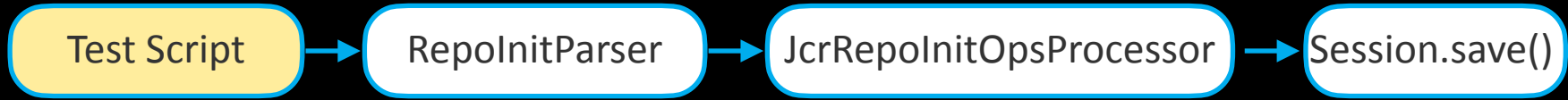
Access Control Lists (ACL)

```
set principal ACL for principall,principal2
  allow jcr:all on /content
  allow jcr:namespaceManagement on :repository
  deny jcr:write on /var/secrets
  remove * on /libs,/apps
end
```

```
set ACL on home(jack),/tmp/a
  allow jcr:read for alice
  deny jcr:read for user2 restriction(rep:itemNames,p1,p2)
end
```

```
delete ACL on /content
delete principal ACL for ada, amy
```

Interlude: how big? how fast?



```
for i in 1 to $LIMIT
do
  create service user LU-$i
  create path /tmp/repointit-three/mod-1/lu-$i
  set ACL for LU-$i
    allow jcr:all on /tmp/repointit-three/mod-1/lu-$i
  end
  set ACL on /tmp/repointit-three/mod-1/lu-$i
    deny jcr:read for everyone
  end
done
```

With limit = 5000:
(20000 repointit operations)

Parsing: 127 msec

Execution: 23868 msec

About 1000 ops / second
on my 2018 macbook pro
on a local Sling Starter instance

Test script at <https://gist.github.com/bdelacretaz/5ece181782206c0c9f820a78e6baaeef>

JCR node types and namespaces

```
register nodetypes
```

```
<<===
```

```
<slingevent='http://sling.apache.org/jcr/event/1.0'>
```

```
<nt='http://www.jcp.org/jcr/nt/1.0'>
```

```
<mix='http://www.jcp.org/jcr/mix/1.0'>
```

```
[slingevent:Event] > nt:unstructured, nt:hierarchyNode
```

```
- slingevent:topic (string)
```

```
- slingevent:application (string)
```

```
===>>
```

```
register namespace (testing) uri:some-uri/V/1.0
```

privileges

```
register privilege simplePrivA  
register privilege ns:simplePrivB
```

```
register abstract privilege abstractA  
register abstract privilege ns:abstractB
```

summary

create path, set properties, default values

create (service) user, delete, disable

create group, add/remove users, delete

set ACL on paths or for principals, remove *

register JCR node types and namespaces

register privileges

repoint vs content packages?

repoint:

diffable

tracable evolution

failures prevent startup
(good for critical setup)

content packages:

complex content structures

project modularization

repoint for repository setup,
content packages for actual content?

WDYT?

The background of the slide is a close-up, high-contrast image of several interlocking metal gears. The gears are dark, with highlights on their teeth and inner surfaces, creating a sense of depth and mechanical complexity. A solid blue horizontal band runs across the middle of the image, containing the text "under the hood!".

under the hood!

The background of the slide is a close-up, high-contrast image of several interlocking metal gears. The gears are rendered in shades of gray and black, with highlights that emphasize their metallic texture and the sharp edges of their teeth. The perspective is slightly angled, giving a sense of depth and mechanical complexity.

writing a parser?

please

DO NOT

write parsers

BY HAND

unless you know

EXACTLY

what you are doing

The repoint parser code is

maintainable
testable
understandable

Thanks to the use of a parser generator (JavaCC)

```
void deleteAclPrincipals(List<Operation> result) :  
{  
    <DELETE> <ACL> <FOR> p = principalsList()  
    ( <EOL> | <EOF> )  
    {  
        result.add(new DeleteAclPrincipals(p));  
    }  
}
```

see also: <https://javacc.org> and

<https://notes.eatonphil.com/parser-generators-vs-handwritten-parsers-survey-2021.html>

please
DO NOT
write parsers
BY HAND
unless you know
EXACTLY
what you are doing

GCC: Handwritten

Source code for the C parser available [here](#). It used to use Bison until [GCC 4.1](#) in 2006. The C++ parser also switched from Bison to a handwritten parser [2 years earlier](#).

Clang: Handwritten

Not only handwritten but the same *file* handles parsing C, Objective-C and C++. Source code is available [here](#).

Ruby: Yacc-like Parser Generator

Ruby uses Bison (thanks for the [correction!](#)). The grammar for the language can be found [here](#).

V8 JavaScript: Handwritten

Source code available [here](#).

Zend Engine PHP: Yacc-like Parser Generator

Source code available [here](#).

TypeScript: Handwritten

Source code available [here](#).

Bash: Yacc-like Parser Generator

Source code available [here](#).

test-driven software

test-driven software

31 commented test scenarios

drive the **design**, **implementation** and **documentation**

(text files, input + expected output)

```
# test scenario
create service user bob,alice, tom21
create service user lonesome
create service user pathA with path some/relative/path
create service user pathA with path /some/absolute/path
```

concatenate

```
# expected parser output
CreateServiceUser bob
CreateServiceUser alice
CreateServiceUser tom21
CreateServiceUser lonesome
CreateServiceUser pathA with path some/relative/path
CreateServiceUser pathA with path /some/absolute/path
```

Appendix A: repoint syntax: parser test scenarios

A concatenation of all test scenarios from the [repoint parser module](#) follows.

Assuming that test suite is complete, this exposes all the language constructs and options, with descriptive comments where needed. If something's unclear, please ask or provide patches for these tests to make them easier to understand.

The following output is generated by the [concatenate-test-scenarios.sh](#) script found in the repoint parser repository.

Repoint parser test scenarios

```
# test-1.txt

create service user bob,alice, tom21
create service user lonesome
create service user pathA with path some/relative/path
create service user pathA with path /some/absolute/path

# test-2.txt

create service user Mark-21
delete service user Leonardo,Winston_32

# test-3.txt

#
# single-word
# We're testing the comments now
# This is A COMMENT with other things like 12, 34
# And now for a tag, <ok> ?
# And some punctuation: .,;:_[]+*%&/()=?^"
# Also with leading whitespace.

# blank lines work, of course
create service user comments_test_passed

# test-4.txt

# trailing comments test
create service user comments_test_passed
# something

# test-5.txt

# trailing comments test without following blank lines
create service user comments_test_passed
# something

# test-10.txt

# Set ACL example from SLING-5355
# Without the "with glob" option, we're not planning to support
# that at this time.
set ACL on /libs,/apps, /, /content/example.com/some-other_path
remove * for user1,user2
allow tcpread for user1 user2
```

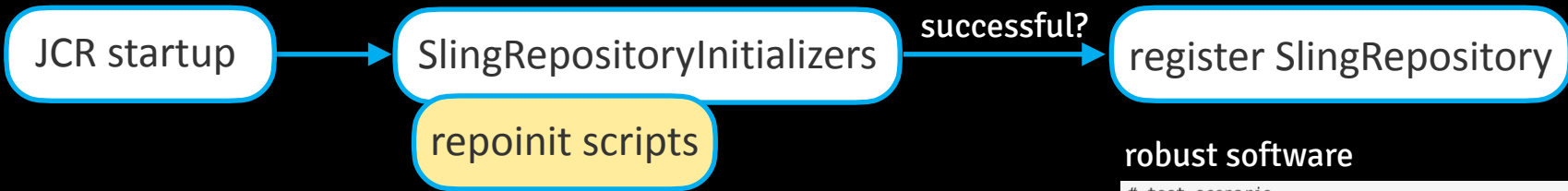
<https://sling.apache.org/documentation/bundles/repository-initialization.html>

The background image shows a person's legs in bright pink rubber boots and another person's legs in dark brown rubber boots, both standing in a muddy, leaf-covered forest floor. The boots are splattered with mud. The scene is captured in a close-up, low-angle shot, emphasizing the texture of the mud and the vibrant color of the boots.

coda

repoint!

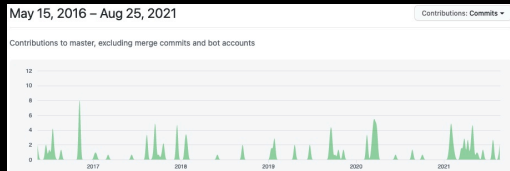
reliable repository initialization



clean syntax

```
set ACL on home(jack),/tmp/a
  allow jcr:read for alice
  deny jcr:read for bob
end
```

ongoing improvements



robust software

```
# test scenario
create service user bob,alice, tom21
create service user lonesome
create service user pathA with path some/relative/path
create service user pathA with path /some/absolute/path
```

# expected parser	output
CreateServiceUser bob	CreateServiceUser bob
CreateServiceUser alice	CreateServiceUser alice
CreateServiceUser tom21	CreateServiceUser tom21
CreateServiceUser lonesome	CreateServiceUser lonesome
CreateServiceUser pathA with path some/relative/path	CreateServiceUser pathA with path some/relative/path
CreateServiceUser pathA with path /some/absolute/path	CreateServiceUser pathA with path /some/absolute/path



I'm @bdelacretaz , thanks for watching!

more at <https://sling.apache.org/documentation/bundles/repository-initialization.html>

on parsing: <https://pinboard.in/u:bdelacretaz/t:parsing/>