Using OSGi for script deployment in Apache Sling

Radu Cotescu, Karl Pauls - Adobe
Who are we?
@raducotescu

- Computer Scientist @ Adobe, Basel, Switzerland
- Member of the Apache Software Foundation
- Apache Sling PMC member
- Maintainer of HTL for Apache Sling
@karlpauls

- Computer Scientist @ Adobe, Basel, Switzerland
- Member of the Apache Software Foundation
- Apache Sling and Apache Felix PMC (VP) member
- Co-Author of OSGi in Action
Do you have a minute to talk about Apache Sling?

- REST-centric web framework, based on an extensible content tree
- JCR for persistence (Apache Jackrabbit Oak)
- Collection of OSGi modules, deployed in Apache Felix
- Powers Adobe Experience Manager
From URLs to Scripts - a simplified view

GET
/hot-drinks/coffee.medium.milk.cup

RESOURCE MAPPING

/.hot-drinks/coffee sling:resourceType=coffee-maker

COLLECT

SCRIPT RESOLUTION

coffee.html
medium.html
medium/milk.html
medium/skimmed-milk.html
large.html
large/milk.html
large/skimmed-milk.html

FILTER & SORT

VALIDATE

medium/milk.html

READ

EVALUATE / COMPILE & EXECUTE

Script Engine
Scripts and Servlets are equal

```java
@Component(service = Servlet.class,
            name = "org.apache.sling.servlets.get.DefaultGetServlet",
            property = {
                "service.description=Default GET Servlet",
                "service.vendor=The Apache Software Foundation",

                // Use this as a default servlet for Sling
                "sling.servlet.resourceTypes=sling/servlet/default",
                "sling.servlet.prefix:Integer=-1",

                // Generic handler for all get requests
                "sling.servlet.methods=GET",
                "sling.servlet.methods=HEAD"
            })
@Designate(ocd=DefaultGetServlet.Config.class)
public class DefaultGetServlet extends SlingSafeMethodsServlet {
}
```
Versioning and dependencies

- There is no standard way of defining either.
- An option would be to use resource type versioning through path conventions.
- Dependencies can only be checked at runtime (but not enforced).
- What happens if your evil colleagues delete a script you were delegating to? Or worse, if they change the whole markup?
Performance

- Each script requires two trips to the persistence layer when first compiled, only to read the script.
- Sling needs to maintain some caches to keep things snappy.
Performance

“There are only two hard things in Computer Science: cache invalidation and naming things.” -- Phil Karlton
Realitiy check

1. What are scripts actually: content or code?
2. Are scripts authored or developed?
3. Can scripts be used freely or do they have constraints?
4. If scripts are code, then why do we treat them differently?
Reality check

Code:

1. provides or implements an API (HTTP in our case)
2. evolves semantically
3. is bundled into a cohesive unit, managed by one or more developers
But what if we...

1. pack scripts into OSGi bundles
2. define the resource types as *versioned capabilities*, with *versioned requirements* (Java APIs, other resource types to which scripts delegate or which scripts extend)
3. allow the platform to do what it’s made to: wire things
Let’s quickly consult the OSGi specification

Capability - Describing a feature or function of the Resource when installed in the Environment. A capability has attributes and directives.

Requirement - An assertion on the availability of a capability in the Environment. A requirement has attributes and directives. The filter directive contains the filter to assert the attributes of the capability in the same Namespace.

https://osgi.org/specification/osgi.core/7.0.0/framework.module.html#framework.module.dependencies
How? Use the Apache Sling Scripting Bundle Tracker[1]

What:
1. add-on module to which bundles that provide scripts have to be wired explicitly
2. reuses the already established mechanisms for registering servlets in Apache Sling
3. allows building light-weight instances that can be thrown into production with very little warm-up, when using precompiled scripts
How? Use the Apache Sling Scripting Bundle Tracker[1]

4. provides the mechanism for deploying truly versionable scripts, with explicit dependencies, by relying on the OSGi framework
5. removes the need of a separate ScriptCache
6. removes additional pressure on the persistence layer
7. simplifies instance and application upgrades
8. there's also a Maven plugin for generating requirements and capabilities
So what’s different?

Option 1: scripts packed as bundle entries

GET
/hot-drinks/coffee.medium.milk.cup

SERVLET
sling.resourceTypes=coffee-maker
sling.selectors=[medium, large, medium.milk, medium.skimmed-milk, large.milk, large.skimmed-milk]
sling.extensions=cup

BUNDLE
javax.script/coffee-maker/1.4.1/
  coffee-maker.html
  medium.html
  medium/milk.html
  medium/skimmed-milk.html
  large.html
  large/milk.html
  large/skimmed-milk.html

Script Engine

SELECT SCRIPT AND
EVALUATE / COMPIL E & EXECUTE

CACHE EXECUTABLE UNIT

SEND RESPONSE

OSGi
Community Event 2019
So what’s different?

Option 2: precompiled scripts

GET
/hot-drinks/coffee.medium.milk.cup

RESOURCE MAPPING

RESOURCES
/hot-drinks/coffee
sling:resourceType=coffee-maker

PUBLIC

SERVLET
sling.resourceTypes=coffee-maker
sling.selectors=[medium, large, medium.milk, medium.skimmed-milk, large.milk, large.skimmed-milk]
sling.extensions=cup

RESOURCE TYPE MAPPING

BUNDLE
coffee_maker/_1_4_1/
coffee_maker.class
medium.class
medium/milk.class
medium/skimmed_milk.class
large.class
large/milk.class
large/skimmed_milk.class

Script Engine

SELECT CLASS AND EXECUTE

CACHE INSTANCE

SEND RESPONSE
How does it work in practice?

**BUNDLE COFFEE-MAKER**

`javax.script/coffee-maker/1.4.1/`
- coffee-maker.html
- medium.html
- medium/milk.html
- medium/skinned-milk.html
- large.html
- large/milk.html
- large/skinned-milk.html

Require-Capability:
- osgi.extender;
  filter="(&(osgi.extender=sling.scripting)(version>=1.0.0)(!(version>=2.0.0)))"

**BUNDLE LATTE-ART-MAKER**

`javax.script/latte-art-maker/1.0.0/`
- latte-art-maker.html

Require-Capability:
- osgi.extender;
  filter="(&(osgi.extender=sling.scripting)(version>=1.0.0)(!(version>=2.0.0)))"
  sling.resourceType;
    filter="(&(sling.resourceType=coffee-maker)&(version>=1.3.0)(!(version>=2.0.0)))"
Sure, but how?

1 Provide-Capability / Script -> 1 Servlet / Script

Provide-Capability

```java
sling.resourceType="latte-art-maker";
        sling.servlet.methods:List<String>="GET";
        version:Version="1.0.0"
```
Demo*  

* or how we can embarrass ourselves if things don’t work
Where does all this lead?

OSGi RFP 196\[2\]

- Provides a way to use an OSGi framework with custom classloaders (a.k.a. OSGi Connect/PojoSR)

Graal/Substrate VM

- Ahead-of-Time (AOT) Java code compilation

Together with the precompiled bundled scripts it should be possible to perform an AOT compilation of a Sling application as a native image\[3\].
Resources

[0] - https://sling.apache.org

Assets licensed from https://stock.adobe.com
Our diagrams were designed with https://whimsical.co/flowcharts
Demo available at https://github.com/raducotescu/eclipsecon-demo
EVALUATE THE SESSIONS

Sign in and vote using the conference app or eclipsecon.org

-1 0 +1