Sling NG Launcher

This page is meant to discuss the requirements and design of a new Sling launcher.

It was created in September 2014 to help build consensus about such a launcher.

Current status

(Updated March 2016)

The current launchpad/runnable jar approach does not seem ideal for modern operations where one wants to drive the creation and configuration of Sling instances from (structured) text files that can be managed in Git or other source code control systems, for programmable infrastructure (aka devops).

The Sling Provisioning Model implements the *Sling Instance Config* concept mentioned below. It is used by Sling Maven plugins to generate runnable launchpad jars as well as (to a certain extent) Karaf features.

The experimental Crankstart launcher was created in early 2014 to support some Sling operations / cluster prototypes: Artyom Stetsenko's prototype and Bertrand's Docker/Sling cluster playground. It was modified in 2015 to support the Sling Provisioning Model and is used in integration tests for the ./contrib /commons/mom/jobs/it module.

Terminology

A *Sling Instance Config* (SIC) fully describes the configuration of a Sling instance, including all bundles, run modes, OSGi configurations, OSGi framework properties, repoinit statements etc.

Sling NG Launcher Design Goals

Must have

- Launch a single, optionally immutable (in terms of its SIC) Sling instance from a SIC.
- Define the SIC in a diff-friendly and structured text format that's also human-friendly, including comments. YAML comes to mind.
- Refer to artifacts using Maven coordinates - these can always be converted to other types of URIs as needed.
- Provide a way to specify digests and/or signatures for those artifacts, to validate them before use.
- Include OSGi configurations and OSGi framework properties (including framework version selection) in the SIC so that everything is defined there
- Allow for merging two or more SICs with well defined rules, optionally using signatures or digests to validate the included SICs
- Allow for run mode dependent settings including artifacts, OSGi configurations and OSGi framework properties
- Provide a way to supply configuration values from the environment, for example to connect the Sling instance to a storage server via an URL supplied at runtime.
- Stay compatible with the current launchpad where it makes sense, in particular for command-line startup options

Nice to have

Some of these need a refined specification to decide whether they are actually needed.

- Use a minimal launcher that "never" needs to change, like Crankstart does.
- Maybe combine declarative instructions (lists of bundles for example) with procedural statements ("start framework") for example to allow for optimized and deterministic startup scenarios.
- Allow for dynamic extensions of the startup mechanism, like Crankstart does.
- If using a runnable jar, allow for influencing the launcher setup (run modes probably) by renaming the runnable jar. That's a useful way of producing runnable jars with multiple modes that are very easy to repurpose, by just renaming.
- Maybe allow for pre-assembling the instance for faster startup, like downloading dependencies in advance and embedding in the launcher jar file or in a "sidecar" jar that can be accessed locally.
- The launcher can output a digest of the overall SIC, to be able to verify that multiple instances have the exact same model.

Open Questions

- Do we need or want run modes in the SIC? We might also create different deployments by combining a set of base SICs in different ways.
- Runtime or build time assembly? Crankstart gets and assembles artifacts at startup time, launchpad at build time.