Sling Jenkins Setup

> **Work In Progress**
> This page is not yet complete, please ask on the dev@sling.apache.org mailing list if anything is unclear or out of date

History and current status

We maintain a (large) number of jobs on the ASF Jenkins instance. Historically, we had a few large reactor builds but those proved to be of limited usefulness:

- the builds were quite slow to complete - 50 minutes on average
- if a module in the reactor failed, the whole build failed

In practice, this meant that the Sling CI jobs were red almost all of the time.

To solve these issues we have decided to create individual jobs for each module. Given that Sling has a large number of modules, it is unrealistic to manage them all manually. As such, we have started using the Jenkins Job DSL Plugin . This allows us to programatically create jobs based on a groovy script.

Views

We currently have the following views defined:

- Sling - holds all Sling jobs
- Sling-Dashboard - holds all Sling jobs that need attention (failed, unstable)

Managing jobs

Since all the job management is now done via a Groovy script manual job creation is discouraged. Altering an existing job is also discouraged since the changes will be undone when the jobs are next regenerated.

Jobs are managed by using a 'seed' job, which processes the DSL script. The job is currently located at https://builds.apache.org/view/S-Z/view/Sling/job/sling-seed-build . Typically there is no need to touch this job.

The script controlling the job management is located at https://github.com/apache/sling-tooling-jenkins/blob/master/create_jobs.groovy . To add a new job, the module must be available on GitHub and listed in the manifest file at https://github.com/apache/sling-aggregator/blob/master/default.xml .

Per-job customisations can be applied by creating a Sling module descriptor in the git repository root, see Sling module descriptor for details.

Pax-Exam tests with SNAPSHOT dependencies

Pax-Exam tests using SNAPSHOT versions are problematic since they have no way of knowing the general Maven context and they don't obey the custom Maven repositories. As such, we need to manually define the Apache SNAPSHOT repository for those tests. One issue were this was done is **SLING-6079** - commons.messaging.mail build fails on Jenkins **RESOLVED** .

If you're using the `org.apache.sling.testing.paxexam` module, this is already incorporated as of version `0.0.3-SNAPSHOT`.

Also, if a Pax-Exam test depends on a SNAPSHOT version of another Sling bundle, the project holding the test will not be rebuilt when a new SNAPSHOT of the included bundle is deployed. This only works if there is a dependency to that bundle and the required version in the pom.xml file.

Testing more complex changes with Docker

If you wish to perform more complex changes, affecting the logic of the script, it's recommended to test these changes on a separate Jenkins instance. This way you minimise the chances that something goes wrong on the ASF Jenkins instance and you get very quick feedback. As of 2016-10-01, a run on sling-seed-build on builds.apache.org takes between 10 and 90 minutes.

The fastest way to setup Jenkins locally (until we fix SLING-6062 - Make it simple to run Jenkins jobs locally OPEN) is to use a local docker instance - https://github.com/bdelacretaz/docker-jenkins-dsl-ready provides a Docker image with all the required plugins (as of 2016-10-19) that you can use as follows:

```
docker build -t jenkins-sling .
export WS=<root of your local sling repo checkout>
docker run -p 8080:8080 -v
$WS/tooling-jenkins:/usr/share/jenkins/ref/jobs/SeedJob/workspace/:ro jenkins-sling
```

And then look at http://localhost:8080/job/SeedJob/ for the seed job execution and http://localhost:8080/ for the list of generated jobs.

That image might not allow you to execute the generated jobs (I haven't tested that so far - patches welcome) but at least verify their generation.

Older Docker test instructions

These instructions allow you to run Jenkins from its official Docker image (which is used as the base image of the above one)

```
docker run -p 8080:8080 -p 50000:50000 -v `pwd`/data:/var/jenkins_home jenkins
```

After configuring Jenkins for the first time, perform the following steps:

1. Install the following plug-ins:
    a. Job DSL plug-in
    b. Javadoc plug-in
    c. Maven plug-in
    d. Test stability plug-in
2. Configure the following Jenkins tools:
    a. JDK 1.7 (latest)
    b. JDK 1.8 (latest)
    c. Maven 3.3.9
3. Create a Freestyle job with a Job DSL build step. You can either have it pull from https://svn.apache.org/repos/asf/sling/trunk/tooling/jenkins, and run the create_jobs.groovy script, or use an inline script for faster prototyping

If you wish to test the actual job execution, label the master with the "Ubuntu" label. Otherwise, only the seed job will run.

Limitations

We rely on the Jenkins Maven Plugin to automatically detect relationship between jobs, e.g. from `org.apache.sling.distribution.core` to `org.apache.sling.distribution.api` . However, this does not take into account the provisioning model. To work around this, we currently set up some very broad rules, such as 'all entries under bundles/ or installer/ trigger a launchpad build'. Also, we manually set some dependencies between some of the launchpad testing jobs. We might revisit this to see if we can do something more fine-grained.

References:

1. Jenkins Job DSL Plugin - https://wiki.jenkins-ci.org/display/JENKINS/Job+DSL+Plugin
2. SLING-6061 - Create per-module Jenkins jobs RESOLVED
3. dev@sling.apache.org - sling ci builds per module : http://markmail.org/message/l6wc4bu5pzw5td3d
4. dev@sling.apache.org - CI alternatives for Sling: http://markmail.org/message/mdn4anwe6kxqxa2z
5. builds@apache.org - Splitting a large build into _many_ smaller builds : http://mail-archives.apache.org/mod_mbox/www-builds/201609.mbox/%3C1472811065.24075.8.camel%40apache.org%3E