Sling Jenkins Setup

- History and current status
- Views
- Managing jobs
- SNAPSHOT parent versions
- Pax-Exam tests with SNAPSHOT dependencies
- Testing more complex changes with Docker
    - Older Docker test instructions
- GitHub API usage
- Validating PRs
- References

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This page is not yet complete, please ask on the dev@sling.apache.org mailing list if anything is unclear or out of date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. We currently use a *GitHub folder* that includes all Sling GitHub repositories with a *Jenkinsfile* present. Adding a file to a repository automatically creates a new job in that folder.

We have previously used the Jenkins Job DSL Plugin, but moved away from it due to the better GitHub integration of the Multibranch Pipeline job type.

Views

We currently have the following entry points defined:

- Sling - GitHub organisation folder that holds all Sling jobs
- Sling-Monitor - View that displays all *master* branch builds that have not completed successfully

Managing jobs

Since all the job management is now done via scripted pipelines script manual job creation is discouraged. Altering an existing job is not possible since the jobs are automatically created and updated.

Job configurations are generated by the Jenkins pipeline library helpers under sling-tooling-jenkins/vars. The recommended `Jenkinsfile` to use in Sling projects is

Jenkinsfile

```
/**
 * Licensed to the Apache Software Foundation (ASF) under one
 * or more contributor license agreements.  See the NOTICE file
 * distributed with this work for additional information
 * regarding copyright ownership.  The ASF licenses this file
 * to you under the Apache License, Version 2.0 (the
 * "License"); you may not use this file except in compliance
 * with the License.  You may obtain a copy of the License at
 *
 *   http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 * software distributed under the License is distributed on an
 * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
 * KIND, either express or implied.  See the License for the
 * specific language governing permissions and limitations
 * under the License.
 */

slingOsgiBundleBuild()
```

The reusable logic is container in classes under sling-tooling-jenkins/src. This allows modules that do not follow the standard job pattern to reuse the main logic (error handling, Sling module descriptor parsing, notifications), while having their own separate stage definitions. The only module which does this currently is the Sling IDE Tooling (see the Sling IDE Tooling Jenkinsfile). It is recommended to first try and apply the default `slingOsgiBundleBuild` build before using non-standard Jenkinsfiles.

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

Changes to sling module descriptors are taken into account in the next build. Changes applied to the Sling pipeline library will take effect in the job after. The reason is that the first build applies the updated settings after it's complete, so the next build will see the updates.

SNAPSHOT parent versions

Jenkins can usually not resolve parent pom SNAPSHOT versions. The main reason for that is that the SNAPSHOT repo url is only defined in the ASF parent (being itself a parent of the sling-parent). The `settings.xml` being used on all Jenkins instances does not define that repo url. For a detailed explanation look at **INFRA-15815** - Getting issue details... STATUS .

The following snippet can be added as workaround to the project pom.xml which references a SNAPSHOT parent to be able to build it with Jenkins.

```
  <repositories>
    <repository>
      <id>apache.snapshots</id>
      <name>Apache Snapshot Repository</name>
      <url>https://repository.apache.org/snapshots</url>
      <releases>
        <enabled>false</enabled>
      </releases>
    </repository>
  </repositories>
```

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** - Getting issue details... STATUS .

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

The information below has not yet been updated to use the Jenkinsfile approach, so it will require some manual tweaks to get to the setup we have on the ASF instance

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.

The fastest way to setup Jenkins locally (until we fix **SLING-6062** - Getting issue details... STATUS) 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.

GitHub API usage

The ASF Jenkins instance is configured to use the GitHub API using a Personal Access Token. This token allows us to get decent rate limits and also to perform write actions on GitHub - for instance setting a commit's status or adding a pull request comment. Currently the used OAuth scope is `repo`.

For legal reasons the ASF Infra team does not allow usage of bot accounts, so currently the token is tied to Robert Munteanu 's GitHub account. This also has the consequence of showing the avatar of the user that owns it, which may be a bit surprising.

Validating PRs

Even PRs might trigger Jenkins Jobs. The hook used for that is described at GitHub webhook relay service. The event will only fire though for actions being triggered by ASF committers (not by outside contributors).

References

1. Jenkins Job DSL Plugin - https://wiki.jenkins-ci.org/display/JENKINS/Job+DSL+Plugin
2. **SLING-6061** - Getting issue details... STATUS
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
6. Multibranch Pipeline recipies
7. **SLING-7245** - Getting issue details... STATUS