Introduction

The 'autorepo' process runs periodically (every 20 minutes), querying Artifactory for docs and schema zips and uploading them to the autorepo location. There are two related scripts that can automatically update the links to a project's documentation and/or schema. This document describes the different options available and how to set each of them up. Specifically it describes:

- autoln - updates links to docs and schema folder versions
- autoschemaln - updates links to individual schema documents

autoln

The autoln script keeps symbolic links to a project's docs and/or schemas to be kept up to date. For example, if the latest build of Spring Framework is 3.1.2.RELASE it makes it easier for users to refer to the docs using a symlink of current and 3.1.x. When the Spring 3.1.3.RELEASE release is performed, the autoln script can ensure the links are automatically updated so users do not need to update bookmarks.

autoln generated links

Below is a list of supported symbolic links that are generated when using autoln. The links can be enabled for a project's docs and/or schema.

- `current` the latest stable build. This is defined as the largest version number ending in `.RELEASE`
- `current-SNAPSHOT` the most recent snapshot. This is defined as the largest version number ending in `CI-SNAPSHOT` or `BUILD-SNAPSHOT`
- `<major>.<minor>.x` the most recently released version starting with `<major>.<minor>`. This reference can refer to a milestone build (in the event there are no builds that end in `.RELEASE`), but will not refer to a snapshot. For example, to get the most recently released version of 3.1, one could use the link `3.1.x`.
- `<major>.<minor>.x-SNAPSHOT` the most recent snapshot starting with `<major>.<minor>`. For example, to get the most recently snapshot of 3.1, one could use the link `3.1.x-SNAPSHOT`.

If a project uses a different naming convention, then a one-time link pointing to the links updated by the autorepo script can be created. For example, if the project uses ga instead of current, a link named ga pointing to current could be created.

Previewing autoln

Since autoln impacts released resources it is best to preview what autoln will do before activating it on a particular project. This can be done by logging onto static.springsource.org and running the test script. The testlinks.php page can be used to see what autoln would do if it were activated on all projects. To use testlinks.php run the following commands:

```
$ cd /var/www/domains/springsource.org/static/autorepo_script
$ php testlinks.php
...
Link Deleted: /opt/s2/ers/servers/static.springsource.org/htdocs/autorepo/docs/spring-
framework/1.1.x
Link Created: cd /opt/s2/ers/servers/static.springsource.org/htdocs/autorepo/docs/spring-
framework/; ln -fs 1.1.5 1.1.x
...
```

Of course not all of these actions are being performed by the autoln process, because each project needs to activate the autoln script. However, it gives developers the knowledge of what would happen if they did opt in. The output from the test script can be a bit much, but the results can be filtered with relative ease. For example, to view only the Spring Integration results with relative paths using the following command:

```
$ php testlinks.php | grep /spring-integration/ | sed 's/\/.*autorepo\///'
Link Deleted: docs/spring-integration/2.1.x
Link Created: cd docs/spring-integration/; ln -fs 2.1.3.RELEASE 2.1.x
Link Created: cd docs/spring-integration/; ln -fs 2.1.4.BUILD-SNAPSHOT 2.1.x-SNAPSHOT
```

```
Link Created: cd docs/spring-integration/; ln -fs 2.2.0.BUILD-SNAPSHOT current-SNAPSHOT
Link Created: cd docs/spring-integration/; ln -fs 2.1.3.RELEASE current
...
```

Be careful to ensure that the filtering matches the entire project as some projects use different directory names for docs and schemas. For example, Spring Framework uses schema/spring and docs/spring-framework. If this is the case, ensure to update the command appropriately. For example:

```
$ php testlinks.php | grep -E /spring\(/\|-framework\) | sed 's/\/.*autorepo\///'
```

Activating autoln

Once the results of autoln have been verified to work for a particular project, it can activated by placing a marker file named `.autoln` in the corresponding project's schema and/or docs location. For example, to activate autoln for spring-framework's schema and docs one would add the following files:

```
touch /var/www/domains/springsource.org/www/htdocs/autorepo/docs/spring-framework/.autoln
touch /var/www/domains/springsource.org/www/htdocs/autorepo/schema/spring/.autoln
```

If desired, the .autoln marker file can be placed in only one of the locations to activate it for only docs or only schema. Remember it will be about 20 minutes before the script is run again and you see any results.

Linking autoln to autorepo

NOTE If the project is already using autorepo, then there is a good chance this is already set up.

Since each project's documents are located in another location it is also necessary to set up a symlink between the current location and the autorepo location. This will ensure all your documentation and schemas are kept up to date. For example, Spring Framework hosts its documentation at /var/www/domains/springsource.org/static/htdocs/spring-framework/docs, but the autorepo is set up to deploy to /var/www/domains/springsource.org/static/htdocs/autorepo/docs/spring. This means we will want to link the Spring docs folder and the Spring autorepo docs folder.

In the example below, we first create a backup of the existing autorepo/docs/spring folder. We then link the actual Spring docs to the autorepo location. This ensures that autorepo is updating the actual Spring docs location. We could also invert the link, but take care to preserve the project's existing documentation.

```
$ cd /var/www/domains/springsource.org/static/htdocs/autorepo/docs/
mv spring .spring
ln -s ../../spring-framework/docs spring
```

Similar to the documents, the schema should point to the corresponding autorepo location. Depending on the project, these steps may need to be repeated for multiple schema folders. For example:

```
$ cd /var/www/domains/springsource.org/static/htdocs/schema
mv beans .beans
ln -fs ../autorepo/schema/spring/current/beans beans
```

Notice for schemas only the latest version is linked. However, we could utilize the other links (i.e. current-snapshot) if there was the need and a convention was established.

autoschemaln

Note: Some projects may manage this logic in their build. If this is true, you can skip the autoschemaln script.

Some projects require that an unversioned schema file is linked to the latest versioned schema file. For example, spring-beans.xsd links to spring-beans-3.1.xsd. However, when Spring 3.2 is released spring-beans.xsd should point to spring-beans-3.2.xsd.

autoschemaln generated links

Below is a list of supported symbolic links that are generated when using autoschemaln.

- `<schema-name>.xsd` the most recently released version of the schema. The file starts with `<schema-name>` and ends with a version. Note that this may match on multiple files in a single directory.

Previewing autoschemaln

Since this does impact released resources it is best to preview what autoschemaln will do before activating it on a particular project. The testschemalinks.php page can be used to see what autoschemaln would do if it were activated on all projects. To use testschemalinks.php run the following commands:

```
$ cd /var/www/domains/springsource.org/static/autorepo_script
$ php testschemalinks.php
...
Link Deleted:
/opt/s2/ers/servers/static.springsource.org/htdocs/autorepo/schema/spring/3.1.1.RELEASE/aop/spring-
aop.xsd
Link Created: cd
/opt/s2/ers/servers/static.springsource.org/htdocs/autorepo/schema/spring/3.1.1.RELEASE/aop; ln -
fs spring-aop-3.1.xsd spring-aop.xsd
...
```

Of course not all of these actions are being done, because each project needs to activate to the autoschemaln script. However, it gives developers the knowledge of what would happen if they did opt in. The output from the test script can be a bit much, but the results can easily be filtered. For example, to view only the Spring Bean results with relative paths use the following command:

```
$ php testschemalinks.php | grep spring-beans | sed 's/\/.*autorepo\///'
Link Deleted: schema/spring/3.1.1.RELEASE/beans/spring-beans.xsd
Link Created: cd schema/spring/3.1.1.RELEASE/beans; ln -fs spring-beans-3.1.xsd spring-beans.xsd
Link Created: cd schema/spring/3.2.0.M1/beans; ln -fs spring-beans-3.2.xsd spring-beans.xsd
Link Created: cd schema/spring/3.2.0.BUILD-SNAPSHOT/beans; ln -fs spring-beans-3.2.xsd spring-
beans.xsd
Link Created: cd schema/spring/3.2.0.M2/beans; ln -fs spring-beans-3.2.xsd spring-beans.xsd
Link Created: cd schema/spring/3.2.0.BUILD/beans; ln -fs spring-beans-3.2.xsd spring-beans.xsd
```

Activating autoschemaln

Once the results of autoschemaln have been verified for a particular project, it can be activated by placing a marker file named .autoschemaln in the corresponding project's autorepo schema location. For example, to activate autoschemaln for spring-framework's schema one would add the .autoschemaln file as shown below:

```
touch /var/www/domains/springsource.org/www/htdocs/autorepo/schema/spring/.autoschemaln
```

Remember it will be about 20 minutes before the script is ran again and you see any results.