Staging Repository Merge

Add support for temporary repositories used for staging for Archiva

Abstract

Creating a repository where the users can deploy their artifacts and once the testing tasks are completed, a user can deploy the artifact to the common place where we can use those artifacts as dependencies in POM files or remove the artifact from the temporary repository if it fails during the testing phase.

Proposal

Apache Archiva is a repository management system where we can deploy our artifacts to a common location and later on we can make use of those artifacts (jar files, war files, zip files, ear files, etc.) in our projects by adding the artifacts as dependencies in the POM file. But here there is no separate place to deploy the artifacts which are are not tested yet and also the artifacts of an ongoing project(modules).So we need to deploy them to the common location where the official artifacts are deployed. So there can be some issues due to the use of those artifacts as dependencies in our projects since they are not properly tested.

So here my idea is to create a separate place (repositories) where the registered user can deploy the staging artifacts. Those artifacts are not visible to common users. Once the testing task is completed (ready for public deployment), we can either merge them to the common place where others can make use of those artifacts or remove them from the staging area (if it fails).

Description

First I would like to give a brief introduction about my approach to adding the staging repository feature.

Adding a Staging Repository

Currently, Archiva has two permanent repositories called internal and the snapshots. Here, my idea is to propose an intermediate staging repository to deploy artifacts which have not been tested yet. It is the Archiva user's task to create this permanent staging repository and attach it for a selected repository. The following example will illustrate that scenario.

Let's say the user has a repository called “data_repo” and he needs to get the support of staging repository. Then he has to create staging repository instance called “data_repo_stage” and he can attach this stage repository to his main repository called “data_repo”. (I hope to offer this function through the web interface). Here, user can add more than one stage repository so that simultaneous releasing function will be supported. Promotion is done through the following path.

data_repo_stage >>> data_repo

I think in the stage repository we should limit the access. Nobody should allow to download the artifacts in the staging repositories other than the developers, QA people, Archiva admin user, and the the guy who have the permission for the promotion. Here, for the QA, promotion and the developers, I am going to use current Archiva roles (Repository Manager and Repository Observer) and assign the permission for them.

I think by default it is the repository manager's task to create the stage repositories and attach it to the main repository that he want to add the staging functionality. But those permissions should be configured to any role according to the requirements.

Creating Roles for this staging repository

Read and Write access to the repository

Here, registered users can read and write artifacts to the repository, where registered users mean developers and the QA people. Here, developers should have both read and write access while the QA guys have only read access. Also QA guys should be able to mark the artifact as success one or a failure one. Then it will be useful for the merging part to take a correct decision.

Promotion of the Artifacts (merging to the common place)

This is the most important part in this project. Once the testing tasks are done, this role is responsible for merging artifacts from staging repository to the internal repository or the snapshot repository. It depends on the artifact(snapshot or a version). Here I am going to consider the following things regarding the merging the artifacts:

- If the artifact that is going to be deployed is not an existing one (new one), just deploy it with creating new meta data file. (User should be able to merge the artifact by clicking the merge button and then should receive a success/failure message from the Archiva side)
- If the artifact is an existing one (in the case of re-release), we need to provide following options:
  - When the user clicks merge button (in the case of re-release), it should notify the user about the existing version in the release repository. (eg : version number + updated date).
  - Then we need to suggest to him the following options:
    - skip the re-release
    - merge the artifact while removing the older one. Here we should only allow the user to remove the latest artifact. Meanwhile, metadata updating should also be done.

In above, I have mentioned only the success scenario of an artifact. There can be a fail scenario as well. If an artifact is failed then the user should manually delete the artifact

Implementation

- add stage repository function so that user can create a stage repository and attach it to a particular release repository. (This functionality is granted only for the admin level users). Proper UI should be provide to the user.
- restrict this staging repository from being used as a plugging repository from out user as mentioned in the pom.xml files
- implement the read action and the write actions and assign them to existing roles. (Repository Manager and Repository Observer)
- implement the promotion role. (Only the merging based on the assumption that a new artifact is being deployed. Not an updated version)
- implement a searching algorithm to check the older version of the artifact which have already been deployed.
- add searching feature (action) to the promotion role and do the merging part by providing the options regarding the version as mentioned above (with a proper UI). Here, metadata updating should be done.
- implement logs for staging repository

Further enhancements

- add the simultaneous deployment functionality for the stage repositories. (Here, need to find a way to check with repository is busy with a deployment)