Local repository separation

Context

The current local repository is a single file structure, stored typically in an individual user's home directory.

The suffers from the following problems:

- there is no locking, so if multiple Maven instances attempt to run on the same machine they can corrupt each other's metadata
- it serves multiple purposes - it is both a cache of remote repository artifacts, and a place to locally install artifacts that you build. Because of this, it is possible that the local cache does not always accurately reflect the state of the remote repository
    - downloading a snapshot from a remote repository also writes the chosen version out as -SNAPSHOT, meaning that continues to get used even if the snapshot repository is removed
    - downloading an artifact from a remote repository with a fixed version does not write metadata, so if that repository is later removed the artifact is still used though a clean build would fail. This particularly affects testing staged releases
- it can be different to isolate differences in the local repository without deleting the entire cache, requiring time consuming downloads.
- it isn't possible have multiple checkouts of the same development version and build them independently (particularly important for CI servers).
- it isn't possible to easily clean out a subsection of the repository
- the artifact code is over-complicated to implement the logic for sharing the storage

Related proposals: Maven 2.1 Artifact Resolution Specification

Solution

General Considerations

This solution aims to not change the current behaviour, other than to make it easier/possible to correct things considered known bugs as documented above. Resolution behaviour should not otherwise be affected and any such changes should be in the related proposal.

This proposal simply alters the storage of the artifacts.

Locking

Locking should be implemented at the individual artifact level. This can be done with a lockfile in the artifact top level directory (rather than the individual version), locking both the metadata and artifact.

An artifact operation should be done with files in a temporary location, and moved to the final location in one operation, wrapped by the creation of the lockfile. This makes the duration of the lock relatively short, so that Maven can simply block on the existence of a lockfile (both read and write operations), and remove it after a short period of time.

Local repository separation

The structure of the local repository should become:

```
.
|-- cache
|   |-- snapshots
|   `-- releases
|-- repositories
|   |-- apache.snapshots
|   |-- central
|   |-- codehaus.snapshots
|   `-- ...
`-- workspace
    |-- default
    |-- workspace1
    `-- ...
```

The purposes of these directories are as follows:

cache (snapshots, releases)	immutable artifacts downloaded from a remote repository. No metadata is stored in this directory tree. Snapshots are stored separately by default to make it easier to remove them periodically.
remote	contains a directory for each remote repository (by repository identifier). This contains the metadata and mutable artifacts from that repository. Metadata files will return to the format `maven-metadata.xml` instead of the current `maven-metadata-<id>.xml` file format. Files in these repositories will typically be snapshots and metadata for releases, since actual releases are not mutable and can be stored in the `cache` directory, however if you wish to store actual artifacts in this directory they will be used instead of the cache, allowing a simple way to rsync a remote repository and use it as a local repository (and vice-versa).
workspace	contains a directory for each local workspace, with the primary one being `default`. This contains the metadata and files for any artifact built by maven (both snapshots, and releases).

Under each of these locations, the standard layout remains as it is now:

```
.
|-- cache
|   |-- snapshots
|   |   |-- com
|   |   |   `-- example
|   |   |       `-- ...
|   |   `-- org
|   |       |-- apache
|   |       |   `-- ...
|   |       `-- codehaus
|   |           `-- ...
|   `-- releases
|       `-- org
|           `-- apache
|               `-- ...
|-- repositories
|   |-- apache.snapshots
|   |   `-- org
|   |       `-- apache
|   |           `-- ...
|   |-- central
|   |-- codehaus.snapshots
|   |   `-- org
|   |       `-- codehaus
|   |           `-- ...
|   `-- ...
`-- workspace
    |-- default
    |   `-- com
    |       `-- example
    |           `-- ...
    |-- workspace1
    `-- ...
```

Search sequence

As current behaviour is to be retained when correct, the solution should aim to merge the metadata across the current workspace and active remote repositories to decide what artifact to use. The artifact can then be utilised from either the workspace, remote, or cache repositories.

Existence in the `cache` directory is not a decision point for using an artifact - this must be achieved and the artifact from there used if possible. This will help enable better utilising the remote repository metadata for tracking the source of an artifact in the future to resolve some of the problems listed in the context section of this proposal.

Workspaces

A CLI parameter `--workspace` should be added to allow switching workspaces. This is just a small portion of a more complete workspace proposal, but the identifier can later be reused if this is implemented first so is compatible. It may also be added to settings.xml in the future, though out of scope for this proposal.

Deployment and Merging

Content will be deployed from the workspace directory, but will not be merged to other sections - there should be no need to migrate any data among the repository sections.

Rolling back a reactor build

While this would be a separate feature, and not the default behaviour, it would now be possible to use a temporary workspace to build artifacts during a reactor build and merge them into another workspace on completion, making an entire reactor build "atomic" with respect to the local repository.

Co-existence with Maven 2.0.x

A best practice should be to change your Maven 2.0.x configuration to use `~/.m2/repository/cache/releases` as the local repository, and move the existing content, as this will co-exist properly with Maven 2.1. And share content for all releases (though snapshots will be installed in different locations).

Upgrade path

There is no need to upgrade existing local repositories - first use of Maven 2.1 will only mean users need to rebuild any local software, as remote artifacts will be redownloaded (however see above for the minimisation of this).

Shared Local Repositories

Though locking will now make this possible, it is still not a recommended practice to share the local repository. However, this structure will allow people to share the `cache` location safely to reduce disk usage if desired.

Settings

The above structure is a default, stored in the `localRepository` setting. However, `snapshotLocalCache`, `releaseLocalCache` and `workspacesDirectory` settings will be added to allow flexible location of those directories.

Implementation Considerations

It is worth reviewing whether JSR-170 could be used to assist in managing the content, particularly in regard to locking - however it should not be used unless it can be mapped to the directory structure and individual files as laid out above.

Votes

+1	brett
0	
-1	

References

- http://docs.codehaus.org/display/MAVEN/Local+repository+separation