PersistenceManager based Backup

Your Rating: Results: 107 rates

Rationale

PersistenceManager is the lowest layer of API between Java and backing database. This approach offers possibility of fast backup/restore since it skips all consistency and boundary checks. It is also severely limiting amount of memory used for backup/restore.
The disadvantage of the approach is the fact that current API was not designed with this possibility in mind and is not as easily accessible as necessary for smooth implementation.

Outline

Try to develop functionality to perform backup/restore using ItemState}}s and {{PersistenceManager.load() and PersistenceManager.store() operations.

Obstacles

o This approach tries to tap on low level JR API and biggest problem is thus obtaining references to the current PM instance to execute such operations.
o Other problems are related - dealing with internal structures of JR, version dependency since accessed code is not part of public API

Code snippets

Access PM at runtime:

```
Method[] ms = getRepo().getClass().getDeclaredMethods();
Method m = null;
for (Method x : ms) {
    if (x.getName().equals("getWorkspaceInfo")) {
        m = x;
        break;
    }
}
m.setAccessible(true);
Object workspaceInfo = m.invoke(s2.getRepository(), "default");
ms = workspaceInfo.getClass().getDeclaredMethods();
m = null;
for (Method x : ms) {
    if (x.getName().equals("getPersistenceManager")) {
        m = x;
        break;
    }
}
m.setAccessible(true);
PersistenceManager pm = (PersistenceManager) m.invoke(workspaceInfo, null);
```