rSmart Kuali Development Process Technical Proposal

Leo Przybylski

July 12, 2012

Contents

1 Goals **2**
1.1 Development 2
1.1.1 Commits 3
1.1.2 Configuration 3
1.1.3 Building 3
1.1.4 Testing 4
1.2 Configuration Management 4

2 Use Cases **4**
2.1 Project Creation 4
2.2 New Prototype 5

2.3 Environment Setup 5
2.4 Start Work 6
2.5 Commit Change 7
2.6 Two or More Issues 8
2.7 Liquibase Changes 9
2.7.1 Creating a Change 9
2.7.2 Testing a Change 10

3 Setup **15**
3.1 Project Structure 15
3.2 Conventions 16

Abstract

This document is intended to complement the Kuali Development process. Outlined below are technical assumptions made about the process and how it will be carried out to simply things for developers, managers, and administrators.

1 Goals

The development process has some specific goals. Really, we had some specific questions we wanted answered and we want to only have to answer them once. We don't want to have to keep asking these questions. DRY (Do not repeat yourself). If we can answer these questions and find a way that we will not have to answer them anymore, then we have simplified our development process a great deal.

1.1 Development

One of the areas we want to simplify is development. Kuali has standardized on Maven who's principal tenet is "Convention over Configuration." Basically, by defining standards and common practices across domains, we can minimize overhead. Let's analyze areas on our current KFS development projects where we have separation instead of standards within the domain.

1.1.1 Commits

The standard is to use the Jira issue name with a commit, so the change can be tracked from Jira. There are two problems we currently have here. With new developers, we always have to emphasize the importance of this. This is one area we always repeat ourselves. Also, sometimes a jira issue is not created for a certain task. Rather than create an issue, the developer completes the task.

Another problem is with multiple checkins. Sometimes with a change, a developer does not do it in one checkin and things get missed. A developer then has to go back and grab the other files that are missing. We keep emphasizing the importance of complete checkins, but human error always interferes. How do we keep from repeating ourselves here?

1.1.2 Configuration

Not to be confused with Configuration Management, this is rather project configuration per-developer. In KFS, we rely a lot on project configuration being similar. Unfortunately, with multiple databases, project-specific dependencies, etc..., it is difficult to maintain a common configuration. Not to mention, if developers are working on more than one project at a time, how do we stay on top of project configurations?

Another issue with configurations is that developers do not follow the same standards. They tend to follow the way they are taught, but not all developers have the same teacher. This leads to project/directory structures and even deveopment styles do differ. Having inconsistent configurations can cause confusion and maintenance hassles. How do we stop repeating ourselves here?

1.1.3 Building

This is partially to do with configuration, but the biggest problem is the length at which builds take, the numerous steps to produce a clean build, and the need to build often. This has been a problem since the beginning of Kuali. Many have solved this problem, but not many are practicing the solutions. We need to start following these practices at rSmart to mitigate the time we spend hassling with builds.

1.1.4 Testing

We boast a lot about TDD, but we have not been practicing what we preach. Is it a matter of discipline? Or habit? How do we keep asking ourselves this and solve it once and for all? Many times, institutions ask us for our input/contribution on automated testing, but we constantly have to build from scratch because we have nothing to begin with.

1.2 Configuration Management

- Consistent configurations for all platforms
- Manage configurations in VCS
  - allows common config to be maintained by revision
- Frequent releases with changes coupled to a Jira version
- Better process management tool integration
- No work without Jira issue

2 Use Cases

Some use cases I put together to illustrate ideal solutions that answer some of the questions above. Ideally, we want to answer them all, but these are based on assumptions I can think of.

2.1 Project Creation

Creating a new KFS project for example:

Listing 1: Create a new project with maven

```
$ mvn rsmart-kfs:create-overlay -DinstitutionId=edu.stevens
-DjiraProject=STEVENS -DprototypeId=com.rsmart.kuali.kfs:kfs:5.0
```

The above requires

- Jira project named stevens created. It contains information in its description in YAML format which the mvn tool will use to generate a new project using the rsmart-kfs archetype.
- Since it is creating an overlay, this implies that in rsmart's maven repository, there exists a prototype with the GAV com.rsmart.kuali.kfs:kfs:5.0

- We are defining this for the stevens institution, so it will create a new project with the groupId edu.stevens.kuali.kfs and pullover the necessary configuration information from the prototype

2.2 New Prototype

Creating a new prototype. A prototype is what is used to create overlays on top of. This is usually base code. I expect most of the prototypes to be either from the org.kuali.kfs group or the com.rsmart.kuali.kfs group.

Listing 2: Deploy a new Prototype

```
$ mvn rsmart-kfs:create-prototype -DgroupId=com.rsmart -
    DprototypeId=com.rsmart.kuali.kfs:kfs:5.0
```

A prototype doesn't need a jira project. In fact, it probably doesn't even get checked into SVN. It is simply a war that is built and deployed to the maven repository specifically for overlays to be built on top of it.

2.3 Environment Setup

From the path where you want your environment created:

Listing 3: Create a new development environment with maven

```
$ mvn rsmart-kfs:setup-development -DjiraId=STEVENS -
    DdeveloperId=lprzybylski
```

This will go and grab the necessary information from the jira project and build a local development for the lprzybylski user. This includes checking out the project and setting up configuration. The configuration is exclusive from other environments/projects. For example, stevens does not use config or step on config from cornell. The project information is encapsulated from the path where the above is run; therefore, deleting the path or copying the path will copy/remove all project related data.

Notice the developerId is captured and stored in local project metadata. This is important for simplying tasks mentioned later. The jira id is also added to local project metadata. The above settings can be overridden, but are now used as defaults.

2.4 Start Work

Listing 4: Assign the above issue to me

```
$ mvn rsmart-kfs:assign -DissueId=STEVENS-12
```

Assigns the given issue. This is not meant to replace the Jira UI at all. Rather, it is intended to use jira more ubiquitously. The problem we are attempting to solve is that there's 3 tools for change management we are trying to merge. Jira, SVN, and Maven. Instead of referring, to SVN revisions and tracking things by SVN revision, we can now track things by Jira issue id.

By executing the above, the following will happen:

1. Lookup Jira Issue
2. Check Jira issue for attached changelist.txt
3. Create changelist.txt if necessary
4. Assign the issue to self

Notice it uses something called a *changelist.txt*. I want to draw attention to this because a changelist is going to become instrumental in managing changes acrossed jira issues. It is explained better in The Red Bean Book. Basically, there will be a new path called changelists/ attached to the project. A changelist will be prefixed with the jira issue. For example, STEVENS-12-changelist.txt. This changelist will contain a list of all the files modified for this Jira issue. It will be helpful because of switching between Jira issues, what is lost is what files were modified. True, a developer could look at the svn commits, but it would be much easier to contain these within the Jira issue. Whenever a jira issue is assigned, the developer can attain the changelist immediately within eclipse. See Subclipse changelist support. Basically, if a changelist exists for an issue, a developer can get ahold of all the files modified in that jira issue through the changelist.

Changelists are also helpful with cherrypicking (see later).

2.5 Commit Change

A user can commit a blanket change like this:

Listing 5: Commit a change to an issue

```
$ mvn rsmart-kfs:commit -DissueId=STEVENS-12
Give a Message:
 --      --    ------    ------
/\ \  - \ \ /\   -- \ /\   -- \
\ \ \/ ".\-\\-\-\/\-\\-\----\
-\-\--/".~\-\\ \-----\\ \-\ \-\
  \/-/    \/-/ \/-----/ \/-/\/-/

 ------   --   --    --    ------
/\--   -\/\ \-\ \ /\ \ /\   ---\
\/-/\ \/\ \   -- \\ \ \\ \---  \
   \ \-\ \ \-\ \-\\ \-\\/\-----\
    \/-/  \/-/\/-/ \/-/ \/-----/

 ------    --   --    ------        --      --    ------    ------
/\   -- \ /\ "-.\-\-/\------\------/\-\----\-\-/\------\-/\------\
\-\-\/\-\\-\-\-.---\\-\------\------\-\-\/-".\  \\ \   -- \\ \---  \
 \ \-----\\ \-\"\-\\-\-----\-----\--/".~\-\\ \-\ \-\\/\-----\
  \/-----/ \/-/ \/-/ \/-----/     \/-/   \/-/ \/-/\/-/ \/-----/

 ------    ------    ------    --        --        --  --
/\  == \ /\   ---\ /\   -- \ /\ \      /\ \      /\ \-\ \
\ \   --< \ \   --\ \ \   -- \\ \ \----\ \ \----\ \---- \
 \ \-\ \-\\ \-----\\ \-\ \-\\ \-----\\ \-----\\/\-----\
  \/-/ /-/ \/-----/ \/-/\/-/ \/-----/ \/-----/ \/-----/

 --   --    ------    ------    -----
/\ \-\ \ /\   -- \ /\  == \ /\   --.
\ \   -- \\ \   -- \\ \  --< \ \ \/\ \
 \ \-\ \-\\ \-\ \-\\ \-\ \-\\ \----.
  \/-/\/-/ \/-/\/-/ \/-/ /-/ \/----/
```

Or itemize change per file:

Listing 6: Commit a change to an issue

```
$ mvn rsmart-kfs:commit -DissueId=STEVENS-12 -Ditemize=true
Do you want to add the following files to STEVENS-12-changelist.
    txt?
M             2618
work/src/org/kuali/kfs/sys/context/PropertyLoadingFactoryBean.
    java

y(N): y

Do you want to add the following files to the project?
?                    build/war
?                    KFSMI-8503.diff

y(N): n
```

```
Give a Message:

* file1:
Creating new configuration for blah

* file.xml
Adding database structure

* file2.xml
Creating a new parameter for blah
```

2.6 Two or More Issues

One common practice that is being used for change management is the concept of cherry picking (and). This is used to be able to add/remove changes from a release in an agile way. We are going to need this very much. One best practice to help facilitate cherry picking is one issue at a time with very few commits. The trouble with this practice is that there is always an occassion when an issue becomes blocked and a developer cannot just sit and wait. It is inevitable there will be a need to work on 2 issues at once. Working on 2 issues at the same time that modify the same file make cherry picking virtually impossible. How can we resolve this?

When using the following command (also mentioned above)

Listing 7: Assign the above issue to me

```
$ mvn rsmart-kfs:assign -DissueId=STEVENS-12
You are currently working on STEVENS-15. You also have
    uncommitted
changes. Before working on STEVENS-12, would you like to stash
    your
work for later? y(N): y
Creating
https://svn.rsmart.com/svn/kuali/kfs/rsmart_kfs_stevens/stashes/
    leo/STEVENS-12...done.
Checking out https://svn.rsmart.com/svn/kuali/kfs/
    rsmart_kfs_stevens/stashes/leo/STEVENS-12...done.
Committing to https://svn.rsmart.com/svn/kuali/kfs/
    rsmart_kfs_stevens/stashes/leo/STEVENS-12...done.
Switching to
https://svn.rsmart.com/svn/kuali/kfs/rsmart_kfs_stevens/trunk...
    done.

You are now on trunk at revision 2618. STEVENS-12 is still
    assigned to
you, but no longer "In Progress". To switch back use:
mvn rsmart-kfs:switch -DissueId=STEVENS-12
```

Maven will check if the current user has any issues in progress. If there are any issues in progress, it will attempt to:

1. Create/merge a branch for the stash.
2. Set the old Jira issue to no longer be "In Progress".
3. Switch back to trunk.
4. Change the new jira issue to be "In Progress".

It is important to know that also switching between issues will result in the same. The difference between assign and switch is that a jira issue will change assignment with assign and not with switch.

2.7 Liquibase Changes

In the past, creating a database change was a hassle. In some cases, multiple database platforms needed to be supported and it was difficult to duplicate SQL. Copy-paste was rampant and led to numerous build failures. Even with build failsafes in place, there were problems with almost every database change because there was no way to test beforehand.

2.7.1 Creating a Change

To create a database change, do the following:

Listing 8: Create a change in liquibase

```
<customChange class="org.liquibase.change.ext.
    CreateResponsibility">
    <param name="template"  value="Review" />
    <param name="namespace" value="KFS-TEM" />
    <param name="name"      value="Review" />
    <param name="active"    value="Y" />
</customChange>
\begin{lstlisting}

Or...
\begin{lstlisting}[language=xml,caption={Add a system parameter
    }]
<customChange class="org.liquibase.change.ext.AddSystemParameter
    ">
    <param name="application" value="Y" />
    <param name="namespace" value="KFS-TEM" />
    <param name="name" value="ENABLE_PER_DIEM_LOOKUP_LINKS_IND"
        />
    <param name="detailTypeCode" value="TravelAuthorization" />
    <param name="typeCode" value="CONFG" />
    <param name="description" value="Y" />
```

```
    <param name="value" value="Y" />
    <param name="constraintCode" value="Y" />
</customChange >
```

Normally, these are very database specific issues, but with some custom refactorings we can remove that and make it this easy. Even within projects, some may require their own refactorings that can be added. This will be especially useful when developing test data for integration tests.

2.7.2 Testing a Change

Once I have created a liquibase change, I will need to test it somehow. If my project supports multiple platforms, I should test against them all. Here is how we would do it. First setup the tests.

```
mvn validate testResources
```

You may recall that the changelogs are copied during the validate goal. I run testResources to copy my properties files to the appropriate locations. After doing that, I should see this in target/changelogs/update/

```
leo@behemoth ~/.workspace/kfs/release-4-0-overlay
(21:38:19) [540] ls target/changelogs/update/CM-156.xml
```

I see the changelog I created. Good. I should also see my properties in target/test-classes/liquibase

```
leo@behemoth ~/.workspace/kfs/release-4-0-overlay
(22:02:22) [541] ls target/test-classes/liquibase/
TEM.properties              TEMNIGHTLY.properties
liquibase.properties.template
```

There you have it. Now we're ready to test.

```
 leo@behemoth ~/.workspace/kfs/release-4-0-overlay
(21:29:10) [537] mvn validate lb:test
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------


[INFO] Building kfs 4.0M2
[INFO] ------------------------------------------------------------------------


[INFO]
[INFO] --- maven-resources-plugin:2.5:copy-resources (copy-test-
    changelogs) @ kfs ---
[debug] execute contextualize
```

```
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 1 resource
[INFO]
[INFO] --- lb-maven-plugin:0.0.1:test (default-cli) @ kfs ---
[WARNING] Artifact with no actual file, 'org.kuali.kfs:kfs'
[WARNING] Artifact with no actual file, 'commons-lang:commons-
    lang'
[WARNING] Artifact with no actual file, 'com.lowagie:itext'
[WARNING] Artifact with no actual file, 'jasperreports:
    jasperreports'
[WARNING] Artifact with no actual file, 'org.kuali.kfs:kfs'
[WARNING] Artifact with no actual file, 'mysql:mysql-connector-
    java'
[WARNING] Artifact with no actual file, 'junit:junit'
[WARNING] Artifact with no actual file, 'javax.servlet:servlet-
    api'
[WARNING] Artifact with no actual file, 'javax.servlet:jstl'
[WARNING] Artifact with no actual file, 'taglibs:standard'
[WARNING] Artifact with no actual file, 'javax.servlet:jsp-api'
[WARNING] Artifact with no actual file, 'org.eclipse.jetty:jetty
    -deploy'
[WARNING] Artifact with no actual file, 'org.eclipse.jetty:jetty
    -jsp-2.1'
[WARNING] Artifact with no actual file, 'org.eclipse.jetty:jetty
    -server'
[WARNING] Artifact with no actual file, 'org.eclipse.jetty:jetty
    -webapp'
[WARNING] Artifact with no actual file, 'org.hamcrest:hamcrest-
    library'
[WARNING] Artifact with no actual file, 'org.springframework:
    spring-beans'
[WARNING] Artifact with no actual file, 'org.springframework:
    spring-context'
[WARNING] Artifact with no actual file, 'org.springframework:
    spring-context-support'
[WARNING] Artifact with no actual file, 'org.springframework:
    spring-core'
[WARNING] Artifact with no actual file, 'org.springframework:
    spring-jdbc'
[WARNING] Artifact with no actual file, 'org.springframework:
    spring-tx'
[WARNING] Artifact with no actual file, 'org.springmodules:
    spring-modules-ojb'
[INFO] Parsing Liquibase Properties File
[INFO]    File: /Users/leo/.workspace/kfs/release-4-0-overlay/
    target/test-classes/liquibase/TEM.properties
[INFO]
    ------------------------------------------------------------------------

[INFO]
    ------------------------------------------------------------------------

[WARNING] Artifact with no actual file, 'org.kuali.kfs:kfs'
[WARNING] Artifact with no actual file, 'commons-lang:commons-
    lang'
[WARNING] Artifact with no actual file, 'com.lowagie:itext'
```

```
[WARNING] Artifact with no actual file, 'jasperreports:
    jasperreports'
[WARNING] Artifact with no actual file, 'org.kuali.kfs:kfs'
[WARNING] Artifact with no actual file, 'mysql:mysql-connector-
    java'
[WARNING] Artifact with no actual file, 'junit:junit'
[WARNING] Artifact with no actual file, 'javax.servlet:servlet-
    api'
[WARNING] Artifact with no actual file, 'javax.servlet:jstl'
[WARNING] Artifact with no actual file, 'taglibs:standard'
[WARNING] Artifact with no actual file, 'javax.servlet:jsp-api'
[WARNING] Artifact with no actual file, 'org.eclipse.jetty:jetty
    -deploy'
[WARNING] Artifact with no actual file, 'org.eclipse.jetty:jetty
    -jsp-2.1'
[WARNING] Artifact with no actual file, 'org.eclipse.jetty:jetty
    -server'
[WARNING] Artifact with no actual file, 'org.eclipse.jetty:jetty
    -webapp'
[WARNING] Artifact with no actual file, 'org.hamcrest:hamcrest-
    library'
[WARNING] Artifact with no actual file, 'org.springframework:
    spring-beans'
[WARNING] Artifact with no actual file, 'org.springframework:
    spring-context'
[WARNING] Artifact with no actual file, 'org.springframework:
    spring-context-support'
[WARNING] Artifact with no actual file, 'org.springframework:
    spring-core'
[WARNING] Artifact with no actual file, 'org.springframework:
    spring-jdbc'
[WARNING] Artifact with no actual file, 'org.springframework:
    spring-tx'
[WARNING] Artifact with no actual file, 'org.springmodules:
    spring-modules-ojb'
[INFO] Parsing Liquibase Properties File
[INFO]    File: /Users/leo/.workspace/kfs/release-4-0-overlay/
    target/test-classes/liquibase/TEM.properties
[INFO]
------------------------------------------------------------------------

[INFO] Executing on Database: jdbc:mysql://localhost:3306/TEM
[INFO] Tagging the database
INFO 9/21/11 9:29 PM:liquibase: Successfully acquired change log
     lock
INFO 9/21/11 9:29 PM:liquibase: Reading from `DATABASECHANGELOG`
INFO 9/21/11 9:29 PM:liquibase: Successfully released change log
     lock
[INFO] Doing update
INFO 9/21/11 9:29 PM:liquibase: Successfully acquired change log
     lock
INFO 9/21/11 9:29 PM:liquibase: Reading from `DATABASECHANGELOG`
INFO 9/21/11 9:29 PM:liquibase: ChangeSet /Users/leo/.workspace/
    kfs/release-4-0-overlay/target/changelogs/update/CM-156.xml::
    CM-156-1::kuali (generated) ran successfully in 37ms
INFO 9/21/11 9:29 PM:liquibase: Successfully released change log
     lock
```

```
[INFO] Doing rollback
INFO 9/21/11 9:29 PM:liquibase: Successfully acquired change log
     lock
INFO 9/21/11 9:29 PM:liquibase: Rolling Back Changeset:/Users/
    leo/.workspace/kfs/release-4-0-overlay/target/changelogs/
    update/CM-156.xml::CM-156-1::kuali (generated)::(Checksum:
    3:85a5e658332342fba8b60df3a29fc393)
INFO 9/21/11 9:29 PM:liquibase: Successfully released change log
     lock
INFO 9/21/11 9:29 PM:liquibase: Successfully released change log
     lock
[INFO]
------------------------------------------------------------------------

[INFO]
[INFO] Parsing Liquibase Properties File
[INFO]    File: /Users/leo/.workspace/kfs/release-4-0-overlay/
    target/test-classes/liquibase/TEMNIGHTLY.properties
[INFO]
------------------------------------------------------------------------

[INFO]
------------------------------------------------------------------------

[WARNING] Artifact with no actual file, 'org.kuali.kfs:kfs'
[WARNING] Artifact with no actual file, 'commons-lang:commons-
    lang'
[WARNING] Artifact with no actual file, 'com.lowagie:itext'
[WARNING] Artifact with no actual file, 'jasperreports:
    jasperreports'
[WARNING] Artifact with no actual file, 'org.kuali.kfs:kfs'
[WARNING] Artifact with no actual file, 'mysql:mysql-connector-
    java'
[WARNING] Artifact with no actual file, 'junit:junit'
[WARNING] Artifact with no actual file, 'javax.servlet:servlet-
    api'
[WARNING] Artifact with no actual file, 'javax.servlet:jstl'
[WARNING] Artifact with no actual file, 'taglibs:standard'
[WARNING] Artifact with no actual file, 'javax.servlet:jsp-api'
[WARNING] Artifact with no actual file, 'org.eclipse.jetty:jetty
    -deploy'
[WARNING] Artifact with no actual file, 'org.eclipse.jetty:jetty
    -jsp-2.1'
[WARNING] Artifact with no actual file, 'org.eclipse.jetty:jetty
    -server'
[WARNING] Artifact with no actual file, 'org.eclipse.jetty:jetty
    -webapp'
[WARNING] Artifact with no actual file, 'org.hamcrest:hamcrest-
    library'
[WARNING] Artifact with no actual file, 'org.springframework:
    spring-beans'
[WARNING] Artifact with no actual file, 'org.springframework:
    spring-context'
[WARNING] Artifact with no actual file, 'org.springframework:
    spring-context-support'
[WARNING] Artifact with no actual file, 'org.springframework:
    spring-core'
```

```
[WARNING] Artifact with no actual file, 'org.springframework:
    spring-jdbc'
[WARNING] Artifact with no actual file, 'org.springframework:
    spring-tx'
[WARNING] Artifact with no actual file, 'org.springmodules:
    spring-modules-ojb'
[INFO] Parsing Liquibase Properties File
[INFO]    File: /Users/leo/.workspace/kfs/release-4-0-overlay/
    target/test-classes/liquibase/TEMNIGHTLY.properties
[INFO]
------------------------------------------------------------------------

[INFO] Executing on Database: jdbc:oracle:thin:@heisenberg.
    rsmart.com:1521:KFS
[INFO] Tagging the database
INFO 9/21/11 9:29 PM:liquibase: Successfully acquired change log
     lock
INFO 9/21/11 9:29 PM:liquibase: Reading from DATABASECHANGELOG
INFO 9/21/11 9:29 PM:liquibase: Successfully released change log
     lock
[INFO] Doing update
INFO 9/21/11 9:29 PM:liquibase: Successfully acquired change log
     lock
INFO 9/21/11 9:29 PM:liquibase: Reading from DATABASECHANGELOG
INFO 9/21/11 9:29 PM:liquibase: ChangeSet /Users/leo/.workspace/
    kfs/release-4-0-overlay/target/changelogs/update/CM-156.xml::
    CM-156-1::kuali (generated) ran successfully in 205ms
INFO 9/21/11 9:29 PM:liquibase: Successfully released change log
     lock
[INFO] Doing rollback
INFO 9/21/11 9:29 PM:liquibase: Successfully acquired change log
     lock
INFO 9/21/11 9:29 PM:liquibase: Rolling Back Changeset:/Users/
    leo/.workspace/kfs/release-4-0-overlay/target/changelogs/
    update/CM-156.xml::CM-156-1::kuali (generated)::(Checksum:
    3:85a5e658332342fba8b60df3a29fc393)
INFO 9/21/11 9:29 PM:liquibase: Successfully released change log
     lock
INFO 9/21/11 9:29 PM:liquibase: Successfully released change log
     lock
[INFO]
------------------------------------------------------------------------

[INFO]
[INFO]
------------------------------------------------------------------------

[INFO] BUILD SUCCESS
[INFO]
------------------------------------------------------------------------

[INFO] Total time: 28.920s
[INFO] Finished at: Wed Sep 21 21:29:49 MST 2011
[INFO] Final Memory: 11M/262M
[INFO]
------------------------------------------------------------------------
```

```
leo@behemoth ~/.workspace/kfs/release-4-0-overlay
(21:29:49) [538]
```

3 Setup

Much of the above implies configuration has been made in the local *$HOME/.m2/settings.xml* to include the following information:

- SVN credentials
- Jira credentials
- Jira URL

3.1 Project Structure

Part of simplifying the development process is a convention/standard of project structure. Most of this will be implied by the maven project archetype.

Path	Description
settings.xml	Project-specific maven settings. This is a maven convention. This settings.xml will get checked in with the project. It is not the same as *$HOME/.m2/settings.xml*.
.m2/repository	Project-specific maven repository. Projects will no longer share a maven repository. Each project/environment will get its own local repo. This will prevent repos from interfering for developers.
scripts/changesets/latest/	Liquibase scripts for building a database from scratch
scripts/changesets/latest/cst/	Liquibase scripts for building a database from scratch. Constraints.
scripts/changesets/latest/dat/	Liquibase scripts for building a database from scratch. Data.
scripts/changesets/latest/idx/	Liquibase scripts for building a database from scratch. Indexes.
scripts/changesets/latest/seq/	Liquibase scripts for building a database from scratch. Sequences.
scripts/changesets/latest/tab/	Liquibase scripts for building a database from scratch. Tables.
scripts/changesets/latest/vw/	Liquibase scripts for building a database from scratch. Views.

Path	Description
scripts/changesets/update/	scripts is the normal path for database scripts. I added a subset to imply a liquibase changeset. The naming convention is JIRA-ISSUE-111-svnrevision.xml. svn revision is included in the name because there can be numerious changes to an issue. These need to be independent of the issue itself unless each change strictly implied a new jira issue. That is just too unreasonable.
changelists/	not to be confused with liquibase changesets, these are the files which are part of the jira issue and files that are impacted by the release.
it/	Maven projects should be modular including a module for integration tests
web/	Maven projects should be modular including a module for web
core/	Maven projects should be modular including a module for core code

3.2 Conventions

I cannot think of any I'd like to add at this point