BuildProcess

Note: *if you just want to try out Isis then there's no need to built it first; just use our quickstart maven archetype*

Introduction

The following assumes that you've set up your development environment; see SettingUpDevelopmentEnvironment if not.

Isis is built using Maven, so you should then be able to get going using two commands:

- `mvn clean install`
- `mvn clean install -D modules=release`

The first command builds all the modules that make up the framework (and will take a good while to run), while the second builds the 'release' module (a POM that defines a consistent set of versions of modules that run together).

However, there is a little more to it than that, because we do have a number of profiles which allow the build to be tailored in various ways. Therefore, have a look at the table below, then look at the various use cases listed further down the page.

Where next?

Once you've got the build going, move onto SmokeTest.

Prereqs

If you want to build the Maven site (as opposed to just building the code) then there is one dependency that must be installed manually into your local repo. This is JIMI, which is needed by the docbkx-maven-plugin to embed images into the output PDFs. You can skip this if you'll never intend to build the site (the `mvn site-deploy` command).

The steps are:

- download JIMI from http://java.sun.com/products/jimi/
- download jimi1_0.zip
- unzip
- install the `JimiProClasses.zip` as the JAR:
    - then install: `mvn install:install-file -D groupId=com.java -D artifactId=jimi -D version=1.0 -D packaging=jar -D file=Jimi/JimiProClasses.zip`

Maven Profiles

`-D modules=`	applib	core	runtimes	progmodel	profilestores	security	viewer	release	support	examples	skin	`clean install ?`	`-D build=full ?`
standard	Y	Y	Y	Y	Y	Y	Y						Y
all	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
skin											Y	Y	Y
applib	Y											Y	Y
core		Y										Y	Y
runtimes			Y									Y	Y
progmodel				Y								Y	Y
profilestores					Y							Y	Y
security						Y						Y	Y

viewers							Y					Y	Y
release								Y				Y	Y
support									Y			Y	
examples										Y		Y	Y

Notes:

- most profiles that support `mvn clean install` also support `-D build=full` (basically, also builds Javadoc JARs). The exception is any profile that includes the 'release' module (we hit problems building javadoc for 'release'; it also doesn't really make sense).
- similarly, most profiles supporting `mvn site-deploy` also support `-D site=full` (basically, full reports, Javadoc, JXR, metrics, code coverage etc). The exception again is those profiles including 'release'
- the prereqs to building the site are (a) to build the skin, and (b) to install the JIMI package into the local repo using `mvn install:install-file -D groupId=com.java -D artifactId=jimi -D version=1.0 -D packaging=jar -D file=/path/to/file`
- running `site-deploy` also requires specifying where to deploy to. To deploy locally, use `-D deploy=local`, which will deploy to `/tmp/m2-sites/isis`.

Typical Use Cases

The first time you run these, you'll need to omit the `-o` (offline) flag in order to download plugins and dependencies. Thereafter though the `-o` flag is *strongly* recommended!

Use case	Command
Quick build	`mvn clean install -o`
Build 'release' module (prereq for example apps)	`mvn clean install -Dmodules=release -o`
Build everything	`mvn clean install -D modules=all -o`
Site skin (prereq to building sites)	`mvn clean install -D modules=skin`
Quick build of site + docs, deploy locally	`mvn site-deploy -D modules=standard -D deploy=local -o`
Full build of site + docs, deploy locally	`mvn site-deploy -D modules=standard -D site=full -D deploy=local -o`
Quick build + site + docs for single module (eg, applib)	`mvn clean install site-deploy -D modules=applib -D deploy=local -o`

Deployability?

This page is still a work-in-progress. Although the set of modules that are deployable as the site is clearly defined, I haven't yet figured out how the codebase itself is to be deployed. The main issue is that the archetypes within `support` and also the `release` modules need to be released after the other modules that make up the standard profile. The problem I've realized is that running the mvn-release-plugin will tag all of trunk, whereas we want to run two releases. It therefore might be necessary to move `support/archetypes` and `release` off into their own trunk. We should see what other projects that release archetypes do, though.