Building

Setting Up Your Workspace

There are building instructions in the workspace itself: http://hg.openjdk.java.net/mlvm/mlvm/file/tip/README.txt

Make a directory which will contain both sets of repositories (patches and full sources), and pull everything there.

The sources repository based at http://hg.openjdk.java.net/hsx/hotspot-comp *(This is as of 9/2011; earlier they were based at jdk7/jdk7 and then at bsd-port /bsd-port.)*

pull sources and patches

```
$ mkdir davinci
$ cd davinci
$ hg fclone http://hg.openjdk.java.net/hsx/hotspot-comp sources
$ hg fclone http://hg.openjdk.java.net/mlvm/mlvm patches
```

Then create symbolic links to the patch directories from the corresponding `.hg` directories of the full sources. There is a shell script which will do this automatically, or you can do it by hand.

link patch repos into source repos

```
$ bash patches/make/link-patch-dirs.sh sources patches
+ ln -s ../../../patches/hotspot sources/hotspot/.hg/patches
+ ln -s ../../../patches/jdk sources/jdk/.hg/patches
+ ln -s ../../../patches/langtools sources/langtools/.hg/patches
$ ls -il patches/hotspot/series sources/hotspot/.hg/patches/series
(should be identical files)
```

Since patches are guarded, you need a non-empty guards file in order to apply patches. Select the desired guards in each source repository that has a patch queue. There is a shell script which will distribute the hg command to each patch queue, or you can do it by hand:

select patch guards

```
$ export davinci=$(pwd) guards="buildable testable"
$ sh patches/make/each-patch-repo.sh hg qselect --reapply $guards \
        '$(sh $davinci/patches/make/current-release.sh)'
```

The shell script `current-release.sh` simply digs out the most recent release hash from the source repository; it will be something like `7836be3e92d0`.

For convenience, everything after the `fclone` operations is packaged up in a makefile, which you can invoke on a variety of targets:

makefile targets

```
$ (cd patches/make; gnumake setup) # build links and select guards
$ (cd patches/make; gnumake check) # check for exact patch base revisions
$ (cd patches/make; gnumake force) # force exact patch base revisions
$ (cd patches/make; gnumake unforce) # restore workspaces to tip revisions
$ (cd patches/make; gnumake patch) # actually apply the selected patches
$ (cd patches/make; gnumake build) # run make in each repo's make subdirectory
$ (cd patches/make; gnumake all) # default target is setup check patch build
$ (cd patches/make; gnumake all RELAX_CHECKS=1) # suppress the check
$ (cd patches/make; gnumake all FORCE_VERSIONS=1) # do 'force' instead of 'check'
```

Things to watch out for in the makefile:

- The `check` and `force` targets are hyper-conservative, assuming the patches are worthless unless the base revision matches exactly.
- The `check` target is likely to fail unless the base revision has been forced, but if it passes you can expect the patches to apply cleanly.
- The `build` target is probably oversimplified.

Building

Once patches are applied, build as usual. Build instructions are posted with the OpenJDK Build Group.

TO DO: What are other good references to build advice?

For the most basic JVM enhancements, you may only need to build a Hotspot JVM and apply incremental testing, rather than build a whole JDK/JRE. See the patch document files, or this wiki, for information about incremental testing procedures for specific patch sets.