How to hack in the OpenJDK™ project

David Herron, OpenJDK Quality Lead, Sun Microsystems
Sandeep Konchady, Java Quality, Sun Microsystems

TS-5230

OpenJDK

Java

Sun microsystems

Learn how to fix or change the OpenJDK™, better understand the OpenJDK implementation, and how to submit your changes to the OpenJDK project

GOAL

Agenda

- Why
- Getting and building the code (mercurial, etc)
- Source tree walk-through, and doing builds
- Testing and Compatibility
- Contributing fixes to the OpenJDK project
- Hacking the OpenJDK

Why hack the OpenJDK?

- It's fun (a large intellectually challenging software project)
- It could make your programs run better
- You can fix bugs that are important to you
- Add or remove features (at the risk of incompatibilities)
  - Truly headless OpenJDK
  - Instrument the JDK to trap common mistakes
  - Ditch CORBA! (and: Become nonconformant!)
  - Add SWT!
- Port to new architectures

Demo: Scenegraph in OpenJDK

Agenda

- Why?
- Getting and building the code (mercurial, etc)
- Source tree walk-through, and doing builds
- Testing and Compatibility
- Contributing fixes to the OpenJDK project
- Hacking the OpenJDK

Repositories

Project home: **http://openjdk.java.net**

Details: http://openjdk.java.net/guide/repositories.html

Code: http://hg.openjdk.java.net/jdk7/jdk7

RSS: http://hg.openjdk.java.net/jdk7/jdk7/rss-log

OpenJDK 7: http://download.java.net/openjdk/jdk7/

OpenJDK 6: http://download.java.net/openjdk/jdk6/

Organized as a mercurial “forest” that connects multiple repositories into one. Must install the forest extension. Mercurial uses a decentralized repository paradigm allowing for flexible directed graphs of repositories

Mercurial setup

- http://openjdk.java.net/guide/repositories.html#installConfig
- Mercurial 0.9.4 (or later) plus forest extension
  - In future other extensions may also be required, such as *jcheck* and *defpath*

`# apt-get install mercurial` (or equivalent)

- http://www.selenic.com/mercurial/wiki/index.cgi/ForestExtension

```
$ hg clone http://hg.akoha.org/hgforest/
$ hg fclone http://hg.openjdk.java.net/jdk7/jdk7/
```

Agenda

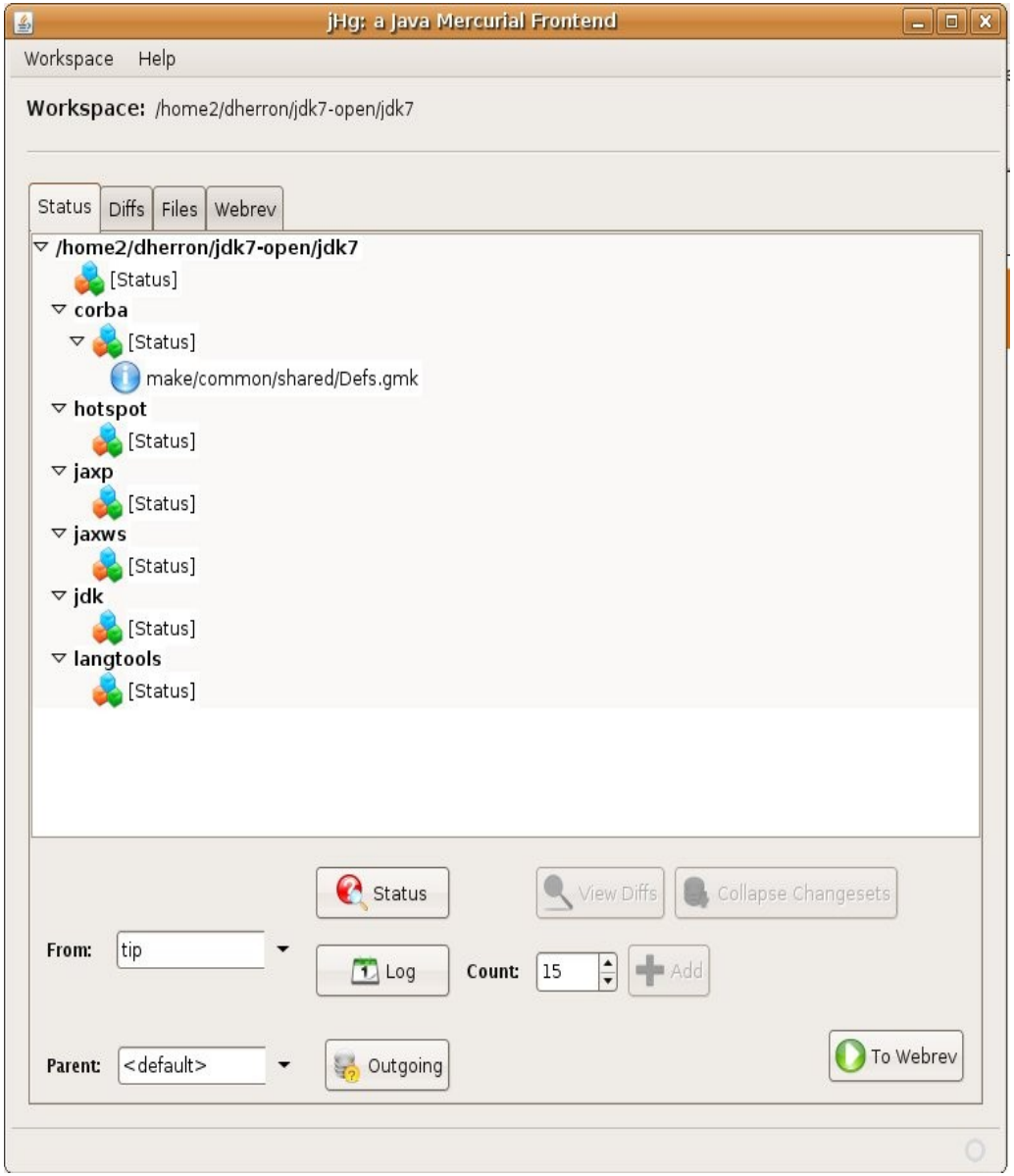
- Why?
- Getting and building the code (mercurial, etc)
- Source tree walk-through, and doing builds
- Testing and Compatibility
- Contributing fixes to the OpenJDK project
- Hacking the OpenJDK

OpenJDK Source tree layout

- Licensing: ASSEMBLY_EXCEPTION, LICENSE, THIRD_PARTY_README
- Build docs: README-builds.html
- Build instructions: Makefile and make subdirectory
- Component repositories: "jdk", "hotspot", "langtools", "corba", "jaxws" and "jaxp"
- Key locations:
  - *comp*/make/Makefile
  - *comp*/src/{share,linux,solaris,...}/{classes,native,...}
  - jdk/make
  - control/make

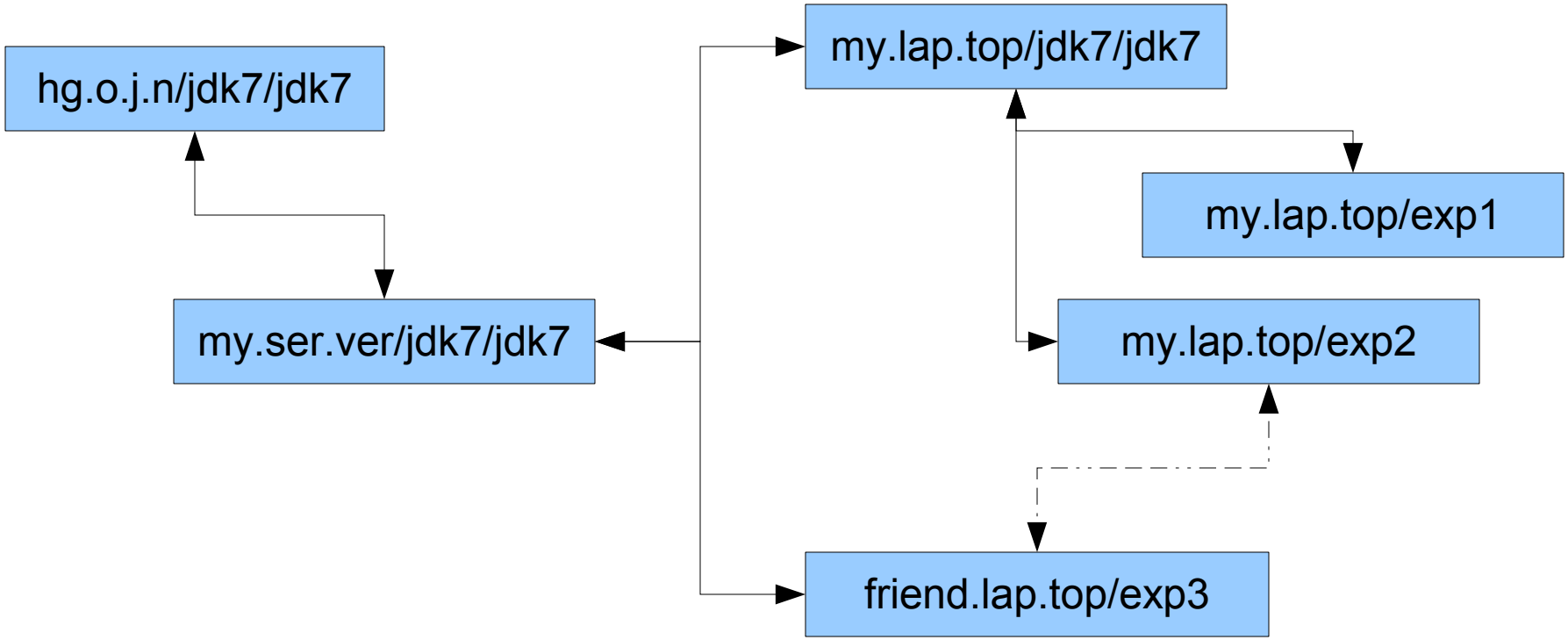
Trees and Forests

- Trees:
  - jdk7/jdk7/{jdk, hotspot, langtools, corba, jaxws, jaxp}
  - Retrieve with: hg clone
- Forest:
  - jdk7/jdk7
  - Retrieve with: hg fclone



Distributed Repositories

- A checked out Mercurial repository is a full repository
  - Allows for a directed graph of repositories
  - Allows for experiments that can be easily thrown away



Build configuration

- Install required libraries & tools
  - See README-builds.html

```
export ALT_BINARY_PLUGS_PATH=..path-to-plugs
export ALT_JDK_IMPORT_PATH=..path-to-Java7-JDK
export ALT_BOOTDIR=..path-to-Java6-JDK
export FINDBUGS_HOME=..path-to-findbugs
export ANT_HOME=...path-to-ant
export OPENJDK=true

. jdk/make/jdk_generic_profile.sh
```

Binary plugs & Boot Java™ Development Kit (JDK™) software & Import JDK software

- The Boot JDK software provides build tools like javac to enable building source
  - Can be previous release (e.g. JDK6 software can bootstrap OpenJDK7 software)
  - With sufficient Makefile hacking other Java™ platform compilers can be used for bootstrap (like GCJ)
- Binary plugs supply the encumbered (non-open source) bits in the OpenJDK
  - OpenJDK incompletely open-sourced
  - Plugs are built from the regular binary JDK software
  - The plugs may go away soon, if encumbrances are cleared
- Import JDK software provides missing pieces in partial builds

Building

- Toplevel Makefile (or **`control/make/Makefile`**):
  - **`make sanity`** – shows whether system configuration is sufficient to perform build
  - **`make all`** – go everywhere, build everything
  - **`make clean`** or **`make clobber`** – clean up a build
  - Results land in **`$(OUTPUTDIR)/j2sdk-image`**
- Each component has a *comp*/make/Makefile
  - Can be run separately from a full build
  - Results land in *comp*/dist

Netbeans™ Projects

- Located in **`jdk/make/netbeans/*`**
  - Additional ones in **`langtools/make/netbeans/*`**
- Allow for full and partial builds
- Allow use of a powerful IDE to edit the OpenJDK
- Make sure to install Netbeans software C/C++ module
- Can run unit tests within these projects (jtreg)
- Pre-configuration required
  - See: **`jdk/make/netbeans/README`**
  - See: **`jdk/make/netbeans/*/README`**

Demo: Netbeans & OpenJDK

Agenda

- Why?
- Getting and building the code (mercurial, etc)
- Source tree walk-through, and doing builds
- Testing and Compatibility
- Contributing fixes to the OpenJDK project
- Hacking the OpenJDK

Testing an OpenJDK build

- Have you broken your OpenJDK build?
- Run demos
- Run your application
- Run big app like Netbeans software
- Test on multiple platforms
- Unit tests:
  - Each workspace has a 'test' directory
  - jtreg is the harness
  - `jtreg -jdk:test-jdk test-or-folder...`
  - http://openjdk.java.net/jtreg/index.html
- Quality group: http://openjdk.java.net/groups/quality

Conformance and the TCK for OpenJDK

- "*Compatibility matters*" is a big value proposition of the Java platform: not just a slogan
  - GPL of course gives freedom to make nonconformant changes
- The Java platform TCK (JCK) is specialized to test conformance with the Java platform specification
  - It's purpose is to find conformance deviations
  - It can also find bugs
- The JCK is available under a specific license to projects substantially derived from the OpenJDK
  - The JCK is not open source
  - Ask on discuss@openjdk.java.net
  - "JCK Acquisition" in OpenJDK Developers Guide
  - http://openjdk.java.net/guide/jckAcquisition.html

Agenda

- Why?
- Getting and building the code (mercurial, etc)
- Source tree walk-through, and doing builds
- Testing and Compatibility
- Contributing fixes to the OpenJDK Project
- Hacking the OpenJDK

Contributing a change to the OpenJDK

- http://openjdk.java.net/contribute/
- Become a contributor: Sign the Sun Contributor Agreement (SCA)
  - Makes clear joint ownership between you and Sun for your contributions
  - Both you and Sun have full rights to the contribution
- Implement a change to the OpenJDK
  - It's good form to first discuss a proposed change, on the relevant OpenJDK mailing list: feedback, advice, etc
- Generate a 'diff' patch and send it to the relevant OpenJDK mailing list
- Find a sponsor from the committers in the relevant group
- Work with your sponsor to evaluate, test and refine the change until it is accepted

Agenda

- Why?
- Getting and building the code (mercurial, etc)
- Source tree walk-through, and doing builds
- Testing and Compatibility
- Contributing fixes to the OpenJDK project
- Hacking the OpenJDK

Demo: Modifying Swing

How to add stuff (Scenegraph, Animation, SWT, Rhino, etc)

- Modify **`control/make/Makefile`**:

```
include ./make/xyzzy-rules.gmk
...
ifeq($(BUILD_XYZZY), true)
  build:: xyzzy
  clobber:: clobber-xyzzy
endif
```

- Add Makefile's:
  - **`control/make/make/xyzzy-rules.gmk`** implements **`xyzzy`** target (for build) and **`xyzzy-clobber`** target
  - Modify **`control/make/make/sanity-defs.gmk`** to have appropriate sanity checks
  - Ensure stuff lands in $(OUTPUTDIR) before final rt.jar etc build

Adding rhino

- Download rhino source `mozilla.org/rhino/download.html`
- Environment variables
  - `export BUILD_RHINO=true`
  - `export RHINO_TOPDIR=..path..to/rhino`
- control/make/Makefile

```
ifeq ($(BUILD_RHINO), true)
  build:: rhino
  clobber:: rhino-clobber
endif
```

control/make/make/rhino-rules.gmk

```
rhino: rhino-build
rhino-build:
ifdef RHINO_TOPDIR
$(CD) $(RHINO_TOPDIR); ant jar
($(CD) $(RHINO_TOPDIR)/build/classes; tar cf - .) | \
 ($(CD) $(OUTPUTDIR)/classes; tar xvf -)
endif

rhino-clobber::
ifdef RHINO_TOPDIR
$(CD) $(RHINO_TOPDIR); ant clean
endif

.PHONY: rhino rhino-build rhino-clobber
```

Demo: More Hacking

For More Information

- Related sessions:
  - OpenJDK Porters' BOF: BOF-6191
  - Future challenges in Open Source and Java™ technology: TS-7064
  - SoyLatte and OpenJDK: Open-Source Java™ Technology for Mac OS X: BOF-7110
  - OpenJDK Community Update: TS-5214
  - The OpenJDK Developers' Guide: BOF-6488
  - OpenJDK Project Q&A: BOF-5215
  - Mercurial Comes to the Netbeans IDE: BOF-5622
  - Who needs Standards in an Open Source World: BOF-5578
- URLs
  - http://openjdk.java.net/guide/
  - http://openjdk.java.net/contribute/

THANK YOU

David Herron, OpenJDK Quality Lead, Sun Microsystems
Sandeep Konchady, Java Quality, Sun Microsystems

TS-5230

How to remove CORBA

- **Will create incompatibility**
  - CORBA is part of the platform spec
  - Removing a feature breaks app's relying on that feature, in other words breaks compatibility
  - Open source means you can do this, and distribute source bundles or binaries that violate the platform spec
- `export BUILD_CORBA=false`
- `WARNING: You are not building the CORBA sources. The corba files will be obtained from the location set in ALT_JDK_IMPORT_PATH.`
- `mv corba corba.ignore`
- Edit: `jdk/make/common/internal/ImportComponents.gmk`
- Edit: `jdk/make/launchers/Makefile`
- There's more to do