Build/Building

JT Harness Build Instructions

In order to contribute code to the project you must have a contributor role. See Mobile & Embedded Community Governance for information about the different community roles.

For a description of how the JT harness repository is organized, read JT Harness Repository Structure.

The JT Harness Developer's Guide describes how the source code is organized and provides important information for developers who want to modify the code.

The JT Harness Source Code Description describes how the JT harness source code is organized.

Note: If you are viewing these instructions on the web site, they apply to the release in development on the repository trunk. At the end of every release cycle, these build instructions are archived with the rest of the web site in the `www/` directory in the tagged branch of the web site that corresponds to that release. If you are building an earlier version of the software please refer to the instructions archived with the tag that corresponds with that release.

Table of Contents

Requirements

The JT harness build requires the technologies listed in the following table.

Technology	Where to Get It	Notes
Ant version 1.6.1 or later	http://ant.apache.org/	Ant must Java 6 or later. Use the `ant -diagnostics` command to verify the version.
ASM Java bytecode manipulation framework v3.1	http://asm.objectweb.org/	Required for compilation only and is not required at run time.
Java SE **version 6** or later	http://java.sun.com/javase/index.jsp	
Java Communications 3.0 API	http://java.sun.com/products/javacomm/index.jsp	Currently, implementations are only available for Solaris SPARC, Solaris x86, and Linux x86. Optional for compilation only and is not required at run time.
JUnit 4.4 library	http://junit.org	JUnit 4.5 has not been tested with JT harness and is therefore not supported. Required for compilation only and is not required at run time.
Servlet libraries (`javax.servlet`)	https://glassfish.dev.java.net	Required for compilation only and is not required at run time. Used for the compilation of include servlet.

Checking out the source

The JTHarness tool repository is located on OpenJDK servers, in the CodeTools Project. It is stored in mercurial vcs, and can be uploaded from mercurial via the following command:

```
% hg clone http://hg.openjdk.java.net/code-tools/jtharness
```

Configure and Run the Build

The following steps describe how to build the JT harness. These instructions assume that your JT harness local working copy is named `JTHarness`.

1. Set up and configure the JDK.

- Download and install the JDK.
- Set the environment variable JAVA_HOME to point to the JDK.

- Set up and configure the Ant software.
- Download and install the Ant software.
- Set your execution path so that it contains the `ant` command. For example: *ANT-dist-path*`/bin/ant`
- Download and extract the remainder of the software listed in the Requirements section.
- Make `JTHarness/build` the current directory.
- Edit the `JTHarness/build/local.properties` file.

- The `BUILD_DIR` property specifies where the build distribution is generated. You may change this as appropriate. If you leave the value undefined, a default location is created under the same parent directory as your working copy.
- Set a property named `jcommjar` to point to the Java Communications API JAR file (typically, `comm.jar`). This is required for compilation **only** if you are using serial communications in your test suite.
- Set a property named `servletjar` to point to the Java EE JAR file (`javaee.jar`). This is required for compilation but is not required at run time. Alternate sources of the `javax.servlet` package can also be used.
- Set a property named `bytecodelib` to include both *path*`/asm-3.1.jar` and *path*`/asm-commons-3.1.jar`. The paths are separated by a colon. This library is used for runtime identification of JUnit tests if required. This is a compile-time requirement, and is not needed at runtime unless you are using that section of the JT harness JUnit libraries.
- Set a property named `junitlib` to point to `junit-4.4.jar`. This library is used to compile the JT harness JUnit library classes packaged in `jt-junit.jar`, not in `javatest.jar`. It is a compile-time requirement, and is not needed at runtime unless you are using that section of the JT Harness JUnit libraries.

  Note: When assigning paths to the properties described in this section, use absolute path names and separate multiple items with a colon.
- Execute the `ant` command (see Additional Build Targets).

The output of the build (build distribution) contains the following two sub-directories:

- `bundles/` -- Contains the generated `jtharness.zip` archive. This Zip archive contains the entire JT harness distribution including documentation, examples, and sample code. Note that the contents of the Zip bundle is extracted into the current directory.
- `binaries/` -- Contains the entire unbundled distribution. It includes the binary and documentation files required to execute the JT harness. The JT harness binary (`javatest.jar`) is generated into the following location:

  ```
  ../../JTHarness-build/binaries/lib/javatest.jar
  ```

Run the JT Harness

You can run the JT Harness in the following ways:

- Specify the "run" target in the Ant build. For more information see Additional Build Targets.
- Run it from the command line:
  `% java -jar` *path*`/javatest.jar`

Refer to the sample test suites in the `examples` directory to see how JT harness can be used in a test suite.

Additional Build Targets

The following additional targets are available. You might wish to identify these targets to your IDE (for example, NetBeans).

Build Target	Description
clean	Removes the entire build distribution directory.
build	Creates and tests the build distribution.
build-examples	Builds the example test suites packaged with the source. This target automatically builds the core harness first.
javadoc	Creates the API documentation.
test	Tests an existing build distribution.
run	Runs the JT harness.