javaROOT

Java to ROOT Interface (M. Lynch, M. Stanitzki, J. Strube)

Introduction

This provides a Java interface to a limited subset of ROOT. The code for this is currently stored on the SVN server (https://heplnm060.pp.rl.ac.uk/repos/javaROOT/trunk). There is also a precompiled version of a later version available, which does not require SWIG. Only the C++ sources then need to be built locally.

This has been tested on

- Scientific Linux Fermi version 4.4, GCC version 3.4.6, Java version 1.6.0.13, ROOT version 5.25 and SWIG version 1.3.40.
- Windows XP SP3, Microsoft C++ Compiler version 15.00.30729.01, MinGW 5.1.4, Java version 1.6.0.14, ROOT version 5.25 and SWIG version 1.3.40.

The libraries are built using a set of makefiles. (On Windows, these are executed using the MinGW port of GNU make, mingw32-make.)

⚠ javaROOT requires SWIG version 1.3.40 or higher, and on Windows, ROOT v.5.25

Features

javaROOT can:

- create histograms (TH1Fs, TH1Ds, TH2Fs, TH2Ds, TProfiles) and fill them with data.

```
sess.newTH1F( "demoA", "demoA", 100, 0, 1 );
for( int i = 0; i < 4096; i++ )
{
        sess.fillTH1F( "demoA", (float) r.nextGaussian() );
}
sess.delete();
```

- create TTrees and fill them with data (branches of type float, int, double, bool, TVector3, TLorentzVector, std::vector<float>, std::vector<int>, std::vector<double>, std::vector<bool>, std::vector<TVector3>, std::vector<TLorentzVector>).

```
sess.newTTree( "demoT", "TestTree", 99 );
sess.branchTTreeFloat( "demoT", "fX" );
for( int i = 0; i < 4096; i++ )
{
        sess.fillBranchFloat( "demoT", "fX", (float) r.nextGaussian() );
        sess.fillTTree( "demoT" );
}
sess.delete();
```

There are a set of example programs demonstrating the available functionality in ./Examples/. These are also built when the main libraries are built with "make all".

There are also a series of tests in ./Tests/ to check that certain basic ROOT functions work correctly on your platform. These are built and run using ./make Tests.

The Java libraries integrate well with Eclipse, with *auto-complete* working out of the box.

Building the Libraries (Linux)

1. Setup the JAVA_HOME environment variable. The `java` executable is supposed to be found in `$JAVA_HOME/bin/java`
2. Setup the ROOT environment: `source <wherever you installed ROOT>/bin/thisroot.sh`
3. Cd to the javaROOT directory.
4. Run `make OSTYPE=linux-gnu all`.
5. Tell the linker where to find the new shared objects. E.g. if running the example "C++", calling export LD_LIBRARY_PATH:$LD_LIBRARY_PATH:. is necessary for the program to run.
6. Running the Java programs requires the classpath to be set appropriately, e.g. java -cp ../../jar/javaROOT.jar:../../jar/freehep-physics-2.1.jar:. Test.
7. If there are any problems with network exceptions on the LCSIM example, it is due to proxy servers: set these when Java is run by adding -Dhttp.proxyHost=<hostname> -Dhttp.proxyPort=<hostport> in the call to Java.

Building the Libraries (Windows)

1. Ensure the %ROOT_SYS% and %JDK_HOME% environment variables are set (go to System Properties (WinKey+Break), Advanced, Environment Variables) to the root folders of ROOT and the JDK respectively.
2. Run a Visual Studio Command Prompt.
3. Cd to the javaROOT root folder.
4. Run mingw32-make all.
5. Running the Java programs requires the classpath to be set approriately, e.g. java -cp ../../jar/javaROOT.jar:../../jar/freehep-physics-2.1.jar:. Test.
6. If there are any problems with network exceptions on the LCSIM example, it is due to proxy servers: set these when Java is run by adding-Dhttp.proxyHost=<hostname> -Dhttp.proxyPort=<hostport> in the call to Java.

Further Details

- javaROOT setup on lxplus
- LCSIM
- Maintenance
- TH1F Example
- TTree Example

Useful Resources

1. http://root.cern.ch/root/html/tutorials/tree/hvector.C.html
2. http://root.cern.ch/phpBB2/viewtopic.php?p=34792&sid=a754a6cf65de5bc9f3af7963e3eb3ffb
3. http://root.cern.ch/drupal/content/interacting-shared-libraries-rootcint
4. http://root-builder.sourceforge.net/instructions.html