POETS Installation & Usage

Tom Hvitved Patrick Bahr Jesper Andersen

Department of Computer Science, University of Copenhagen
Universitetsparken 1, 2100 Copenhagen, Denmark
{hvitved,bahr,jespera}@diku.dk

August 2013

Contents

1 Preliminaries

This document describes how to install POETS from scratch and how to use a running system. In order to install POETS, the following software is required:

- The POETS source code. It is available in the Mercurial repository at https://bitbucket.org/jespera/poets.
- GHC and Cabal. The easiest way to get GHC and Cabal is to install the GHC Platform, avaiable from http://hackage.haskell.org/platform/.
- Thrift. The latest developer version of Thrift can be obtained from http://thrift.apache.org/developers/.

- At least version 6 of the JDK (tested only on the version from Oracle)
- The Android SDK, available from http://developer.android.com/sdk/.
- The Apache Ant tool.

2 The POETS server

Installing the POETS server requires GHC and Thrift.

2.1 GHC

First install the latest version of GHC, including Cabal. After installing GHC, issue the following commands:

```
cabal update
cabal install HTTP
ghc-pkg expose ghc-7.4.1
```

This will install the GHC package HTTP (needed for Thrift), and make the GHC package available (needed in order to compile reports). Note that the version number 7.4.1 depends, of course, on the version of GHC that is installed.

2.2 Thrift

Thrift is needed both for the server and the Android clients communicating with the server. In order to install Thrift, issue the following commands in the Thrift source library:

```
./bootstrap
./configure --without-c_glib --without-csharp --without-java \
 --without-erlang --without-python --without-perl --without-php \
 --without-php_extension --without-ruby --without-haskell \
 --without-go
make
sudo make install
cd lib/hs
cabal install
cd ../java
ant
```

This installs the `thrift` executable for generating e.g. Haskell code from Thrift specification files, as well as Thrift bindings for Haskell and Java. In particular a jar-file `libthrift-x.x.x-snapshot.jar` should now be present in the `lib/java/build/` folder (relative to the main folder of Thrift).

2.3 POETS Thrift library

The POETS Thrift library is a GHC library generated by Thrift from the files in the folder `src/thrift` in the POETS source code folder. It is installed by issuing the following commands in the POETS repository:

```
cd src/haskell/thrift
make
```

2.4 The Server

Finally, the server itself can be installed by issuing the following commands in the POETS repository:

```
cd src/haskell
cabal install
```

This will install the `poetsserver` executable that is the server. Note that the first installation may take a rather long time, since all library dependencies are downloaded and installed as well.

3 Android Client

An Android-based client have been developed for communicating with the POETS server. The client is meant to be run on Android 3.1 or upwards. It has been successfully run on the Samsung Galaxy Tab 10.1 running both Honeycomb and Ice Cream Sandwich the Samsung Galaxy S 2 running Ice Cream Sandwich (although the screen on the phone is not really large enough).

3.1 POETSIO library

The Android client makes use of a small utility library called POETSIO. The source code for POETSIO can be found in `src/java/poetsio/`. One can package the library in a jar-file by running the following command from the folder `src/java/poetsio/`. Before running the command one needs to copy the Thrift Java library as generated during the installation of Thrift described above (`libthrift.jar`) to the `lib` subfolder of the POETSIO folder.

```
ant dist
```

This command will compile the Java classes residing in the `src` subfolder and package them into a jar-file in `src/java/poetsio/dist/`.

3.2 Tablet client

Building To build the Android tablet client follow the below steps:

1. Copy `libthrift.jar` to the `src/java/tablet/libs/` folder.

2. Generate the `poestio.jar` file as described in the previous section and copy it to `src/java/tablet/libs/`.
3. Run `ant debug` from the `src/java/tablet/` folder.

The above steps should produce an file in the `src/java/tablet/bin` folder named `Continuum-debug.apk`.

Installing To install the tablet client onto an Android tablet make sure that you have enabled the ability install apps from non-market sources. (There is probably a toggle somewhere in `Settings>Security`). To put the app onto the device first connect the device to your computer via the usb-connection and then run the following command from the `src/java/tablet/` folder:

```
adb -d install -r bin/Continuum-debug.apk
```

4 Usage and Configuration

4.1 The POETS Server

Building the POETS server as described in Section 2 results in the `poetsserver` executable. The server is started by pointing the executable to a configuration file, e.g. `poetsserver muERP.config`. The configuration file determines the debugging level, the location of the debug log file, and the location of the event log file which is *the only* persistent state of the running system.

The server can be started on an initially empty event log, which results in an "empty system" that only contains predefined data definitions, predefined report functions, and predefined contract functions as described in the tech report [1]. The location of these system definitions is:

```
src/haskell/defs
```

and the contents of these files are not supposed to be changed as part of configuring a running system. Rather the contents of these files can be thought of as a system prelude that is always present in any system. Inspecting these files can however be helpful to see e.g. which event types are present in any system. For example, the file `src/haskell/defs/data/Event.pce` contains the definition

```
Event is abstract.
Event has a DateTime called internalTimeStamp.
```

which defines the super type of all events that are logged in the system. Hence all events that are logged in the event log are known to contain a field called `internalTimeStamp` of the type `DateTime`.

4.2 Interface to the POETS Server

The only interface to a running POETS server is the procedures described in the Thrift specification file. This file is located at `src/thrift/poets.thrift`. For a complete list of procedures see this file.

4.3 Bootstrapping a Fresh System

As described in Section 4.1 it is possible to start the POETS server on an initially empty system. In order to get an actually running system, we must hereafter add data definitions, report definitions, and contract definitions, as described in the tech report [1, Section 3.4]. In order to ease this process, rather than adding the definitions manually by invoking the relevant Thrift procedures described in the previous section, we provide a quick-and-dirty Java program.

The bootstrap program resides in the folder

```
src/java/bootstrap
```

and it is invoked by issuing the command

```
ant boot -Dfile=setup.ini
```

The ant-script runs that Java bootstrap program that reads which definitions to add from a simple named configuration file (here named `setup.ini`) which consists of five parts:

1. Location of ontology files to add.
2. Location of a custom report prelude.
3. Location of report files.
4. Location of a custom contract prelude.
5. Location of contract files.

The ontology files consists of four files, corresponding to the division into the root concepts Data, Transaction, Contract, and Report [1].

The custom report prelude is a means to provide common functionality for report definitions. Since the report engine does not have a proper module system at the time of writing, the custom prelude is simply inlined (HACK!) into each report file before it is uploaded. Note that we cannot solve this problem by moving the custom prelude functions into the system prelude, as these functions typically depend on ontology definitions such as Payment that are not necessarily present in all systems.

The custom contract prelude is similar to that for reports, and the same inlining hack is used until a proper module system is implemented.

References

[1] Tom Hvitved, Patrick Bahr, and Jesper Andersen. Domain-specific languages for enterprise systems. Technical report, Department of Computer Science, University of Copenhagen, 2011.