PiggyBank

Piggy Bank - User Defined Pig Functions

The *Piggy Bank* is a place for Pig users to share their functions. The functions are contributed "as-is". If you find a bug or if you feel a function is missing, take the time to fix it or write it yourself and contribute the changes.

Shared code is in the Apache Pig SVN repo. For APIs see 'contrib: Piggybank' entries in the main Pig Javadoc API pages, eg. 0.8 (somewhat hidden under "Developers", then versioned releases then "Misc.", "API Docs").

Using Functions

To see how to use your own functions in a pig script, please, see the Pig Latin Reference Manual. Note that only JAVA functions are supported at this time.

In brief, you either *DEFINE* a function to give it a short name, or else call it with full package name as shown below.

The functions are currently distributed in source form. Users are required to checkout the code and build the package themselves. No binary distributions or nightly builds are available at this time.

To build a jar file that contains all available user defined functions (UDFs), please follow the steps:

1. Create a directory for the Pig source code: `mkdir pig`
2. cd into that directory: `cd pig`
3. Checkout the Pig source code: `svn checkout http://svn.apache.org/repos/asf/pig/trunk/ .`
4. Build the project: `ant`
5. cd into the piggybank dir: `cd contrib/piggybank/java`
6. Build the piggybank: `ant`
7. You should now see a piggybank.jar file in that directory.

Make sure your classpath includes the hadoop jars as well. This worked for me using the cloudera CDH2 / hadoop AMIs:

```
pig_version=0.4.99.0+10   ; pig_dir=/usr/lib/pig ;
hadoop_version=0.20.1+152 ; hadoop_dir=/usr/lib/hadoop ;
export CLASSPATH=$CLASSPATH:${hadoop_dir}/hadoop-${hadoop_version}-core.
jar:${hadoop_dir}/hadoop-${hadoop_version}-tools.jar:${hadoop_dir}
/hadoop-${hadoop_version}-ant.jar:${hadoop_dir}/lib/commons-logging-1.0.4.
jar:${pig_dir}/pig-${pig_version}-core.jar
export PIG_CONF_DIR=/path/to/mapred-site/and/core-site/pointing/to/your
/cluster
```

To obtain `javadoc` description of the functions run `ant javadoc` from `trunk/contrib/piggybank/java` directory. The documentation is generate in `trunk/contrib/piggybank/java/build/javadoc` directory.

To use a function, you need to figure out which package it belongs to. The top level packages correspond to the function type and currently are:

- org.apache.pig.piggybank.comparison - for custom comparator used by ORDER operator
- org.apache.pig.piggybank.evaluation - for eval functions like aggregates and column transformations
- org.apache.pig.piggybank.filtering - for functions used in FILTER operator
- org.apache.pig.piggybank.grouping - for grouping functions
- org.apache.pig.piggybank.storage - for load/store functions

(The exact package of the function can be seen in the javadocs or by navigating the source tree.)

For example, to use the UPPER command:

```
REGISTER /public/share/pig/contrib/piggybank/java/piggybank.jar ;
TweetsInaug  = FILTER Tweets BY org.apache.pig.piggybank.evaluation.string.
UPPER(text) MATCHES '.*(INAUG|OBAMA|BIDEN|CHENEY|BUSH).*' ;
STORE TweetsInaug INTO 'meta/inaug/tweets_inaug' ;
```

Contributing Functions

For details on how to create UDFs, please, see the UDF Manual. Note that only JAVA functions are supported at this time.

To contribute a new function, please, follow the steps:

1. Check existing javadoc to make sure that the function does not already exist as described in #Using_Functions
2. Checkout UDF code as described in #Using_Functions
3. Place your java code in the directory that makes sense for your function. The directory structure as of now has two levels: function type as described in #Using_Functions and function subtype (like math or string for eval functions) for some of the types. If you feel that your function requires a new subtype, feel free to add one.
4. Make sure that your function is well documented and uses javadoc style of documentation.
5. Make sure that your code follows Pig coding conventions described in HowToContribute
6. Make sure that for each function, you add a corresponding test class in the test part of the tree.
7. Submit your patch following the process described HowToContribute