CSVFiles

2 Feb 2011

CSV Files

2 Feb 2011

1 Introduction

These components allow Comma Separate Values (CSV) files to be imported into SmartFrog deployments. This enables

- Values produced in a spreadsheet to be used in a deployment, with no manual conversion process.
- Values produced by applications which output CSV content to be used directly.
- Bulk data to be fed directly to those components which are designed to read from a tuple data source.

CSV files are an interesting data source, because they are very widely generated. Unfortunately, there is less consistency in format than for XML documents, and so

1.1 Components

1.2 CSVFileToRead

This component reads a CSV File.

```
CSVFileToRead extends File {
    sfClass "org.smartfrog.services.filesystem.csvfiles.CSVFileReadImpl";
    headerLines 0;
    separator ",";
    quoteChar "\"";
    //override some defaults
    mustBeDir false;
    mustRead true;
    testOnStartup true;
    testOnLiveness false;
   /**
    * min number of lines {@value}
    */
    minCount 0;
    /**
    * max number of lines {@value}
    */
    maxCount -1;
    /**
    * minimum width; -1 for do not check {@value}
    */
    minWidth 0;
    /**
    * max width; -1 for do not check {@value}
    */
    maxWidth -1;
}
```

headerLines	Number of header lines to skip
separator	Separator character
quoteChar	Quote Character
minCount	Minimum number of non-header lines in the file
maxCount	Maximum number of lines in the file (or -1 for any number)
minWidth	Minimum width of any row
maxWidth	Maximum width of any row (or -1)

To use this component

1.2.1 CSVColumnReader

This component runs through a CSV file, extracting a single entry from every line, building the result up into a list of string values. This list can be attached to any component.

```
CSVColumnReader extends Prim {
  sfClass "org.smartfrog.services.filesystem.csvfiles.CSVColumnReader";
  sfShouldTerminate true;
  source TBD;
  column 1;
  trimFields true;
  skipEmptyFields true;
  skipNarrowLines false;
  target LAZY THIS;
  targetAttribute "result";
}
```

This component allows content of a CSV file to be used as the input for any SmartFrog component which supports a string list attribute; this list is built by reading in the CSV file and extracting one vertical column.

source	LAZY reference to a CSV File Reader or other tuple data source
column	the column to extract (minimum value: 0)
trimFields	Should strings be trimmed; that is, should leading and trailing white space be removed.
skipEmptyFields	Should empty "" fields be added to the list, or should they be skipped?
skipNarrowLines	If a line is too narrow to have the desired column, should that line be skipped, or should an error be raised
target	A LAZY reference to the component to receive the attribute
targetAttribute	the name of the attribute to update

2 Example

Here is a fragment of the test code; three components which create a CSV file, read it in, then finally parse it to extract a single column of values attached to the parent compound.

```
sfConfig extends Compound {

sourceFile extends TempFileWithCleanup {
  text ##"h1","h2","h3","h4"
"l1e1","","","e4 "
" l2e1","l2 e2","l2'e3,","l2;e4"
this,line,"has, less",spaces
#;
}

reader extends CSVFileToRead {
      filename LAZY sourceFile;
    }

parser extends CSVColumnReader {
      source LAZY action:reader;
      column 1;
      target LAZY PARENT;
  }
}
```

3 Implementation Details

The `CSVFileReadImpl` class implements the `TupleDataSource` interface:

```
package org.smartfrog.services.filesystem;

public interface TupleDataSource extends Remote {
    /**
     * Get the next line
     * @return the next line, all broken up, or null for no new lines.
     * @throws RemoteException network problems
     * @throws SmartFrogException parsing/file IO problems
     */
    String[] getNextTuple() throws RemoteException, SmartFrogException;

    /**
     * Go back to the start of the file
     * @throws RemoteException network problems
     * @throws SmartFrogException parsing/file IO problems
     */
    void start() throws RemoteException, SmartFrogException;

    /**
     * Close the reader. harmless if we are already closed
     * @throws RemoteException network problems
     * @throws SmartFrogException parsing/file IO problems
     */
    void close() throws RemoteException, SmartFrogException;
}
```

This can be implemented by any component which wishes to provide a sequence of string tuples.

When a `CSVFileReader` is deployed, it does not yet load or extract the CSV file's contents, this is postponed until a caller starts the process and then reads the tuples in. This operation is not thread-safe: if two components ask the reader for tuples, they will get different lines from the same read.

Those components that act upon tuple data sources are expected to perform their work in a separate thread; this stops slow data access from interfering with deployment. Once the processing is complete, they can signal a workflow termination.

The actual process of parsing CSV files is delegated to the opencsv library, from http://opencsv.sourceforge.net/.