Carma-cloud

Generated by Doxygen 1.9.4

Chapter 1

CARMAcloud

1.1 Documentation

CARMAcloud provides some of the infrastructure components for CARMA. It enables users to define geofences, rules of practice, replay storms to test weather-related rules of practice, as well as monitor CARMA-enabled vehicles and the messages and controls exchanged with them.

1.2 Deployment

CARMAcloud can be deployed on a Linux server. Ensure you have a properly configured git client and Java Development Kit before executing the following commands:

```
cd /tmp
git clone https://github.com/usdot-fhwa-stol/carma-cloud.git
wget http://apache.mirrors.lucidnetworks.net/tomcat/tomcat-9/v9.0.34/bin/apache-tomcat-9.0.34.tar.gz && tar
       -xzf apache-tomcat-9.0.34.tar.gz && mv apache-tomcat-9.0.34 tomcat && rm -rf
       apache-tomcat-9.0.34.tar.gz
mkdir -p tomcat/webapps/carmacloud/ROOT && mv CARMACloud/web/* tomcat/webapps/carmacloud/ROOT/
find ./CARMACloud/src -name "*.java" > sources.txt && mkdir -p
       tomcat/webapps/carmacloud/ROOT/WEB-INF/classes
javac -cp tomcat/lib/servlet-api.jar:CARMACloud/lib/commons-compress-1.18.jar:CARMACloud/lib/javax.json.jar
       -d tomcat/webapps/carmacloud/ROOT/WEB-INF/classes @sources.txt
sed -i '/<\/Engine>/ i \ \ \ \ \  <Host name="carmacloud" appBase="webapps/carmacloud" unpackWARs="false"
       autoDeploy="false">\n      </Host>' tomcat/conf/server.xml
echo -e '127.0.0.1\tcarmacloud' | sudo -u root tee -a /etc/hosts
mv CARMACloud/lib tomcat/webapps/carmacloud/ROOT/WEB-INF/
touch tomcat/webapps/carmacloud/event.csv
mv CARMACloud/osmbin/rop.csv tomcat/webapps/carmacloud/
mv CARMACloud/osmbin/storm.csv tomcat/webapps/carmacloud/
java -cp tomcat/webapps/carmacloud/ROOT/WEB-INF/classes/:tomcat/lib/servlet-api.jar cc.ws.UserMgr ccadmin
       admin_testpw > tomcat/webapps/carmacloud/user.csv
gunzip CARMACloud/osmbin/*.gz
mv CARMACloud/osmbin tomcat/webapps/carmacloud/
rm -f sources.txt && rm -rf CARMACloud
sudo -u root mv tomcat /opt/
```

These commands will download the CARMAcloud source code from github, necessary dependencies, and the tomcat webserver. Changes to the tomcat version might be necessary if version 9.0.34 is no longer available on the Apache mirror. Next the java code will be compiled and the .class files will be placed in the correct directory. Tomcat's server.xml file will have the carmacloud host entry inserted in the correct location. Carmacloud will be added to the /etc/hosts file. The java command that runs cc.ws.UserMgr will create the ccadmin user for the system. It is suggested to change to password to something more secure by replacing "admin_testpw" with the desired password in the command.

1.3 Configuration

The Tomcat webserver must be configured to run on the deployment server. Click here to find the documentation for Tomcat. There are many configuration items to considered but two that must be addressed for any deployment are adding the domain name and SSL Certificate to the server.xml file. Proper security measures dealing with file permissions should be taken as well. According to Tomcat's documentation a tomcat user and group should be created in the operating system. The standard configuration for file permissions is to have all Tomcat files owned by root with group tomcat. Owner should have read/write permissions, group should have read permission, and world has no permissions. The exceptions are the logs, temp and work directory that are owned by the tomcat user rather than root. Additional users can be added to CARMAcloud by run the following command and replacing <username> and <password> with the desired user name and password respectively:

```
java -cp tomcat/webapps/carmacloud/ROOT/WEB-INF/classes/:tomcat/lib/servlet-api.jar cc.ws.UserMgr <username>
      <password> » /opt/tomcat/webapps/carmacloud/user.csv
```

Additionally, you will need to generate an access token from Mapbox, and replace the text <your access token goes here> with your access token in the /opt/tomcat/webapps/carmacloud/ROOT/script/map.js file. Once everything is configured for the deployment, run the following command to start the application:

```
sudo -u tomcat /opt/tomcat/bin/catalina.sh start
```

1.4 Contribution

Welcome to the CARMA contributing guide. Please read this guide to learn about our development process, how to propose pull requests and improvements, and how to build and test your changes to this project. CARMA Contributing Guide

1.5 Code of Conduct

Please read our CARMA Code of Conduct which outlines our expectations for participants within the CARMA community, as well as steps to reporting unacceptable behavior. We are committed to providing a welcoming and inspiring community for all and expect our code of conduct to be honored. Anyone who violates this code of conduct may be banned from the community.

1.6 Attribution

The development team would like to acknowledge the people who have made direct contributions to the design and code in this repository. CARMA Attribution

1.7 License

By contributing to the Federal Highway Administration (FHWA) Connected Automated Research Mobility Applications (CARMA), you agree that your contributions will be licensed under its Apache License 2.0 license. CARMA License

1.8 Contact

Please click on the CARMA logo below to visit the Federal Highway Adminstration(FHWA) CARMA website. For more information, contact CARMA@dot.gov.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 Package cc.geosrv

Classes

- class Mercator

6.2 Package cc.util

Classes

- class Arrays
- class BufferedInStream
- class CsvReader
- class Geo
- class MathUtil
- class StringPool
- class Text

6.3 Package cc.ws

Classes

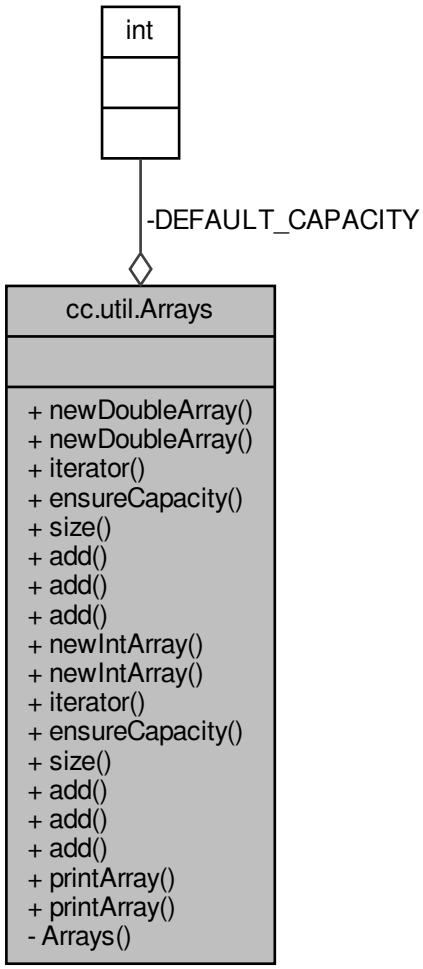
- class EventMgr
- class Handler
- class ReplayMgr
- class RopMgr
- class Session
- class SessMgr
- class User
- class UserMgr

Chapter 7

Class Documentation

7.1 cc.util.Arrays Class Reference

Collaboration diagram for cc.util.Arrays:



Classes

- class DoubleGroupIterator
- class GroupIterator
- class IntGroupIterator

Static Public Member Functions

- static double[ ] newDoubleArray ()
- static double[ ] newDoubleArray (int nCapacity)
- static Iterator< double[ ]> iterator (double[ ] dSrc, double[ ] dDest, int nStart, int nStep)
- static double[ ] ensureCapacity (double[ ] dVals, int nDemand)
- static int size (double[ ] dVals)
- static double[ ] add (double[ ] dVals, double d1)
- static double[ ] add (double[ ] dVals, double d1, double d2)
- static double[ ] add (double[ ] dVals, double[ ] dMore)
- static int[ ] newIntArray ()
- static int[ ] newIntArray (int nCapacity)
- static Iterator< int[ ]> iterator (int[ ] nSrc, int[ ] nDest, int nStart, int nStep)
- static int[ ] ensureCapacity (int[ ] nVals, int nDemand)
- static int size (int[ ] nVals)
- static int[ ] add (int[ ] nVals, int n1)
- static int[ ] add (int[ ] nVals, int n1, int n2)
- static int[ ] add (int[ ] nVals, int[ ] nMore)
- static void printArray (double[ ] dArray, int nStart, PrintStream oPrint) throws Exception
- static void printArray (int[ ] nArray, int nStart, PrintStream oPrint) throws Exception

Private Member Functions

- Arrays ()

Static Private Attributes

- static final int DEFAULT_CAPACITY = 12

7.1.1 Detailed Description

Author

bryan.krueger

Definition at line 11 of file Arrays.java.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 Arrays()

```
cc.util.Arrays.Arrays ( )  [inline], [private]
```

Definition at line 16 of file Arrays.java.

```
00017     {
00018     }
```

7.1.3 Member Function Documentation

7.1.3.1 add() [1/6]

```
static double[] cc.util.Arrays.add (
             double[] dVals,
             double d1 )  [inline], [static]
```

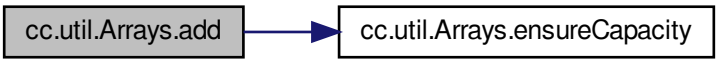
Definition at line 58 of file Arrays.java.

```
00059     {
00060         dVals = ensureCapacity(dVals, 1);
00061         dVals[(int)dVals[0]] = d1; // current insertion position
00062         dVals[0] += 1.0; // update position
00063         return dVals;
00064     }
```

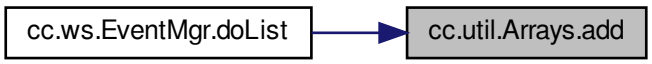
References cc.util.Arrays.ensureCapacity().

Referenced by cc.ws.EventMgr.doList().

Here is the call graph for this function:



Here is the caller graph for this function:



7.1.3.2 add() [2/6]

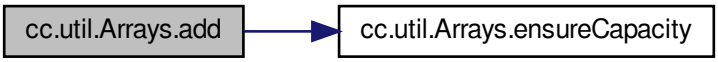
```
static double[] cc.util.Arrays.add (
            double[] dVals,
            double d1,
            double d2 )  [inline], [static]
```

Definition at line 67 of file Arrays.java.

```
00068     {
00069         dVals = ensureCapacity(dVals, 2);
00070         int nIndex = (int)dVals[0]; // current insertion position
00071         dVals[nIndex++] = d1;
00072         dVals[nIndex++] = d2;
00073         dVals[0] = (double)nIndex; // track insertion position
00074         return dVals;
00075     }
```

References cc.util.Arrays.ensureCapacity().

Here is the call graph for this function:



7.1.3.3 add() [3/6]

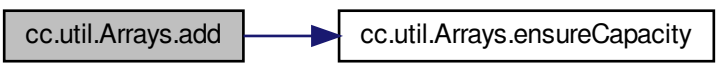
```
static double[] cc.util.Arrays.add (
            double[] dVals,
            double[] dMore )  [inline], [static]
```

Definition at line 78 of file Arrays.java.

```
00079     {
00080         dVals = ensureCapacity(dVals, dMore.length);
00081         int nIndex = (int)dVals[0]; // current insertion position
00082         System.arraycopy(dMore, 0, dVals, nIndex, dMore.length);
00083         dVals[0] = nIndex + dMore.length; // track insertion position
00084         return dVals;
00085     }
```

References cc.util.Arrays.ensureCapacity().

Here is the call graph for this function:



7.1.3.4 add() [4/6]

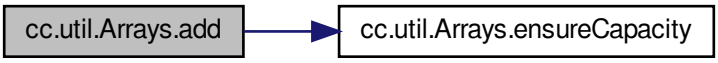
```
static int[] cc.util.Arrays.add (
             int[] nVals,
             int n1 )  [inline], [static]
```

Definition at line 125 of file Arrays.java.

```
00126     {
00127         nVals = ensureCapacity(nVals, 1);
00128         nVals[nVals[0]] = n1; // current insertion position
00129         ++nVals[0]; // update position
00130         return nVals;
00131     }
```

References cc.util.Arrays.ensureCapacity().

Here is the call graph for this function:



7.1.3.5 add() [5/6]

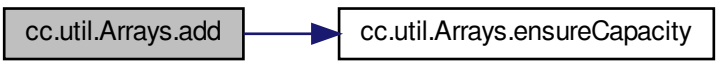
```
static int[] cc.util.Arrays.add (
             int[] nVals,
             int n1,
             int n2 )  [inline], [static]
```

Definition at line 134 of file Arrays.java.

```
00135     {
00136         nVals = ensureCapacity(nVals, 2);
00137         int nIndex = nVals[0]; // current insertion position
00138         nVals[nIndex++] = n1;
00139         nVals[nIndex++] = n2;
00140         nVals[0] = nIndex; // track insertion position
00141         return nVals;
00142     }
```

References cc.util.Arrays.ensureCapacity().

Here is the call graph for this function:



7.1.3.6 add() [6/6]

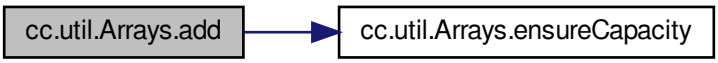
```
static int[] cc.util.Arrays.add (
            int[] nVals,
            int[] nMore )  [inline], [static]
```

Definition at line 145 of file Arrays.java.

```
00146     {
00147         nVals = ensureCapacity(nVals, nMore.length);
00148         int nIndex = nVals[0]; // current insertion position
00149         System.arraycopy(nMore, 0, nVals, nIndex, nMore.length);
00150         nVals[0] = nIndex + nMore.length; // track insertion position
00151         return nVals;
00152     }
```

References cc.util.Arrays.ensureCapacity().

Here is the call graph for this function:



7.1.3.7 ensureCapacity() [1/2]

```
static double[] cc.util.Arrays.ensureCapacity (
            double[] dVals,
            int nDemand )  [inline], [static]
```

Definition at line 41 of file Arrays.java.

```
00042     {
00043         if ((int)dVals[0] + nDemand < dVals.length)
00044             return dVals; // no changes needed
00045 
00046         double[] dNew = new double[nDemand + (3 * dVals.length » 1)];
00047         System.arraycopy(dVals, 0, dNew, 0, (int)dVals[0]);
00048         return dNew;
00049     }
```

Referenced by cc.util.Arrays.add().

Here is the caller graph for this function:

7.1.3.8 ensureCapacity() [2/2]

```
static int[] cc.util.Arrays.ensureCapacity (
            int[] nVals,
            int nDemand )  [inline], [static]
```

Definition at line 108 of file Arrays.java.

```
00109     {
00110         if (nVals[0] + nDemand < nVals.length)
00111             return nVals; // no changes needed
00112 
00113         int[] nNew = new int[nDemand + (3 * nVals.length » 1)];
00114         System.arraycopy(nVals, 0, nNew, 0, nVals[0]);
00115         return nNew;
00116     }
```

7.1.3.9 iterator() [1/2]

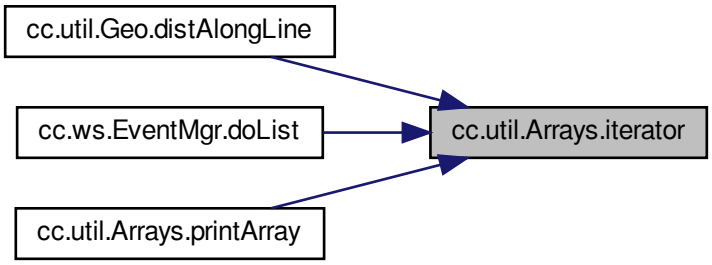
```
static Iterator< double[]> cc.util.Arrays.iterator (
            double[] dSrc,
            double[] dDest,
            int nStart,
            int nStep )  [inline], [static]
```

Definition at line 35 of file Arrays.java.

```
00036     {
00037         return new DoubleGroupIterator(dSrc, dDest, nStart, nStep);
00038     }
```

Referenced by cc.util.Geo.distAlongLine(), cc.ws.EventMgr.doList(), and cc.util.Arrays.printArray().

Here is the caller graph for this function:



7.1.3.10 iterator() [2/2]

```
static Iterator< int[]> cc.util.Arrays.iterator (
            int[] nSrc,
            int[] nDest,
            int nStart,
            int nStep )  [inline], [static]
```

Definition at line 102 of file Arrays.java.

```
00103     {
00104         return new IntGroupIterator(nSrc, nDest, nStart, nStep);
00105     }
```

7.1.3.11 newDoubleArray() [1/2]

```
static double[] cc.util.Arrays.newDoubleArray ( )  [inline], [static]
```

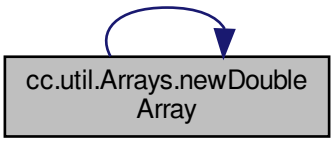
Definition at line 21 of file Arrays.java.

```
00022     {
00023         return newDoubleArray(DEFAULT_CAPACITY);
00024     }
```

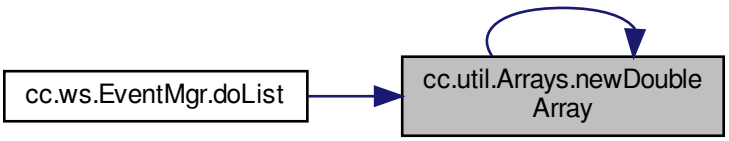
References cc.util.Arrays.DEFAULT_CAPACITY, and cc.util.Arrays.newDoubleArray().

Referenced by cc.ws.EventMgr.doList(), and cc.util.Arrays.newDoubleArray().

Here is the call graph for this function:



Here is the caller graph for this function:



7.1.3.12 newDoubleArray() [2/2]

```
static double[] cc.util.Arrays.newDoubleArray (
            int nCapacity )  [inline], [static]
```

Definition at line 27 of file Arrays.java.

```
00028     {
00029         double[] dVals = new double[++nCapacity]; // reserve slot for size
00030         dVals[0] = 1.0; // initial position is always one
00031         return dVals;
00032     }
```

7.1.3.13 newIntArray() [1/2]

```
static int[] cc.util.Arrays.newIntArray ( )  [inline], [static]
```

Definition at line 88 of file Arrays.java.

```
00089     {
00090         return newIntArray(DEFAULT_CAPACITY);
00091     }
```

References cc.util.Arrays.DEFAULT_CAPACITY, and cc.util.Arrays.newIntArray().

Referenced by cc.util.Arrays.newIntArray().

Here is the call graph for this function:

Here is the caller graph for this function:

7.1.3.14 newIntArray() [2/2]

```
static int[] cc.util.Arrays.newIntArray (
            int nCapacity )  [inline], [static]
```

Definition at line 94 of file Arrays.java.

```
00095     {
00096         int[] nVals = new int[++nCapacity]; // reserve slot for size
00097         nVals[0] = 1; // initial position is always one
00098         return nVals;
00099     }
```

7.1.3.15 printArray() [1/2]

```
static void cc.util.Arrays.printArray (
            double[] dArray,
            int nStart,
            PrintStream oPrint ) throws Exception  [inline], [static]
```

Definition at line 155 of file Arrays.java.

```
00156     {
00157         Iterator<double[]> oIt = iterator(dArray, new double[1], nStart, 1);
00158         boolean bWrite = oIt.hasNext();
00159         if (bWrite)
00160         {
00161             double[] dVal = oIt.next();
00162             oPrint.append(Double.toString(dVal[0]));
00163         }
00164         while (oIt.hasNext())
00165         {
00166             double[] dVal = oIt.next();
00167             oPrint.append(",").append(Double.toString(dVal[0]));
00168         }
00169         if (bWrite)
00170             oPrint.append("\n");
00171     }
```

References cc.util.Arrays.iterator().

Here is the call graph for this function:

7.1.3.16 printArray() [2/2]

```
static void cc.util.Arrays.printArray (
            int[] nArray,
            int nStart,
            PrintStream oPrint ) throws Exception  [inline], [static]
```

Definition at line 174 of file Arrays.java.

```
00175     {
00176         Iterator<int[]> oIt = iterator(nArray, new int[1], nStart, 1);
00177         boolean bWrite = oIt.hasNext();
00178         if (bWrite)
00179         {
00180             int[] nVal = oIt.next();
00181             oPrint.append(Integer.toString(nVal[0]));
00182         }
00183         while (oIt.hasNext())
00184         {
00185             int[] nVal = oIt.next();
00186             oPrint.append(",").append(Integer.toString(nVal[0]));
00187         }
00188         if (bWrite)
00189             oPrint.append("\n");
00190     }
```

References cc.util.Arrays.iterator().

Here is the call graph for this function:

7.1.3.17 size() [1/2]

```
static int cc.util.Arrays.size (
            double[] dVals )  [inline], [static]
```

Definition at line 52 of file Arrays.java.

```
00053     {
00054         return (int)dVals[0];
00055     }
```

7.1.3.18 size() [2/2]

```
static int cc.util.Arrays.size (
            int[] nVals )  [inline], [static]
```

Definition at line 119 of file Arrays.java.

```
00120     {
00121         return nVals[0];
00122     }
```

7.1.4 Member Data Documentation

7.1.4.1 DEFAULT_CAPACITY

```
final int cc.util.Arrays.DEFAULT_CAPACITY = 12  [static], [private]
```

Definition at line 13 of file Arrays.java.

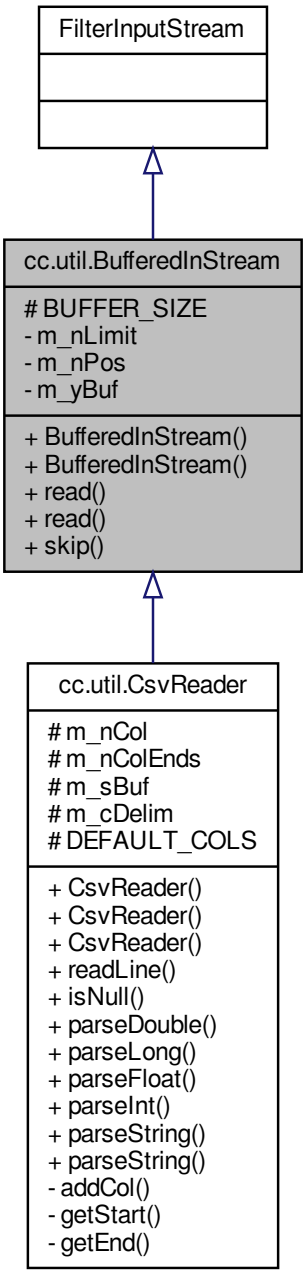
Referenced by cc.util.Arrays.newDoubleArray(), and cc.util.Arrays.newIntArray().

The documentation for this class was generated from the following file:

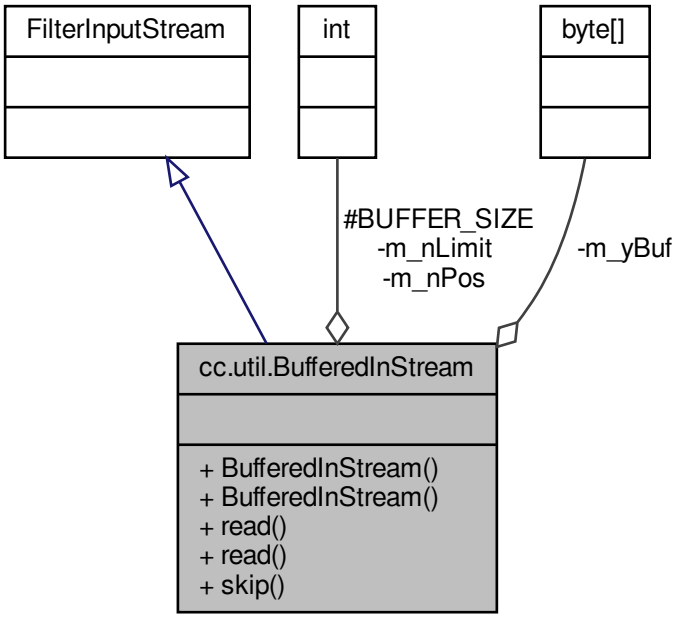
- src/cc/util/Arrays.java

7.2 cc.util.BufferedInStream Class Reference

Inheritance diagram for cc.util.BufferedInStream:



Collaboration diagram for cc.util.BufferedInStream:



Public Member Functions

- BufferedInStream (InputStream oInputStream, int nSize)
- BufferedInStream (InputStream oInputStream)
- int read () throws IOException
- int read (byte[ ] yBuf, int nOff, int nLen) throws IOException
- long skip (long lBytes) throws IOException

Static Protected Attributes

- static final int BUFFER_SIZE = 8192

Private Attributes

- int m_nLimit
- int m_nPos
- byte[ ] m_yBuf

7.2.1 Detailed Description

Definition at line 8 of file BufferedInStream.java.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 BufferedInStream() [1/2]

```
cc.util.BufferedInStream.BufferedInStream (
            InputStream oInputStream,
            int nSize )  [inline]
```

Definition at line 17 of file BufferedInStream.java.

```
00018     {
00019         super(oInputStream);
00020         m_yBuf = new byte[nSize];
00021     }
```

References cc.util.BufferedInStream.m_yBuf.

7.2.2.2 BufferedInStream() [2/2]

```
cc.util.BufferedInStream.BufferedInStream (
            InputStream oInputStream )  [inline]
```

Definition at line 24 of file BufferedInStream.java.

```
00025     {
00026         this(oInputStream, BUFFER_SIZE);
00027     }
```

References cc.util.BufferedInStream.BUFFER_SIZE.

7.2.3 Member Function Documentation

7.2.3.1 read() [1/2]

```
int cc.util.BufferedInStream.read ( ) throws IOException  [inline]
```

Definition at line 31 of file BufferedInStream.java.

```
00033     {
00034         if (m_nPos >= m_nLimit) // check for empty buffer
00035         {
00036             if ((m_nLimit = in.read(m_yBuf, 0, m_yBuf.length)) <= 0)
00037                 return -1; // no bytes to read and/or read failed
00038
00039             m_nPos = 0; // reset buffer read position
00040         }
00041         return ((int)m_yBuf[m_nPos++]) & 0xff;
00042     }
```

References cc.util.BufferedInStream.m_nLimit, cc.util.BufferedInStream.m_nPos, and cc.util.BufferedInStream.m_yBuf.

Referenced by cc.util.CsvReader.readLine().

Here is the caller graph for this function:

7.2.3.2 read() [2/2]

```
int cc.util.BufferedInStream.read (
            byte[] yBuf,
            int nOff,
            int nLen ) throws IOException  [inline]
```

Definition at line 46 of file BufferedInStream.java.

```
00048     {
00049         int nStart = nOff; // save for length calculation
00050         while (nLen > 0) // repeat until request is fulfilled or stream end
00051         {
00052             if (m_nPos >= m_nLimit) // check for empty buffer
00053             {
00054                 if ((m_nLimit = in.read(m_yBuf, 0, m_yBuf.length)) <= 0)
00055                     return nOff - nStart; // no chars to read and/or read failed
00056
00057                 m_nPos = 0; // reset buffer read position
00058             }
00059
00060             int nBytes = Math.min(nLen, m_nLimit - m_nPos); // available bytes
00061             System.arraycopy(m_yBuf, m_nPos, yBuf, nOff, nBytes); // copy buffer
00062             m_nPos += nBytes; // adjust buffer position
00063             nOff += nBytes; // increment dest offset
00064             nLen -= nBytes; // decrement remaining length
00065         }
00066         return nOff - nStart; // return copied byte count
00067     }
```

References cc.util.BufferedInStream.m_nLimit, cc.util.BufferedInStream.m_nPos, and cc.util.BufferedInStream.m_yBuf.

7.2.3.3 skip()

```
long cc.util.BufferedInStream.skip (
            long lBytes ) throws IOException  [inline]
```

Definition at line 71 of file BufferedInStream.java.

```
00073     {
00074         int nAvailable = (m_nLimit - m_nPos);
00075         if (lBytes <= nAvailable)
00076         {
00077             m_nPos += lBytes;
00078             return lBytes;
00079         }
00080
00081         m_nPos = m_nLimit; // skip buffer entirely
00082         return in.skip(lBytes - nAvailable) + nAvailable; // re-include buffer count
00083     }
```

References cc.util.BufferedInStream.m_nLimit, and cc.util.BufferedInStream.m_nPos.

7.2.4 Member Data Documentation

7.2.4.1 BUFFER_SIZE

```
final int cc.util.BufferedInStream.BUFFER_SIZE = 8192  [static], [protected]
```

Definition at line 10 of file BufferedInStream.java.

Referenced by cc.util.BufferedInStream.BufferedInStream().

7.2.4.2 m_nLimit

```
int cc.util.BufferedInStream.m_nLimit  [private]
```

Definition at line 12 of file BufferedInStream.java.

Referenced by cc.util.BufferedInStream.read(), and cc.util.BufferedInStream.skip().

7.2.4.3 m_nPos

```
int cc.util.BufferedInStream.m_nPos  [private]
```

Definition at line 13 of file BufferedInStream.java.

Referenced by cc.util.BufferedInStream.read(), and cc.util.BufferedInStream.skip().

7.2.4.4 m_yBuf

```
byte [] cc.util.BufferedInStream.m_yBuf  [private]
```

Definition at line 14 of file BufferedInStream.java.

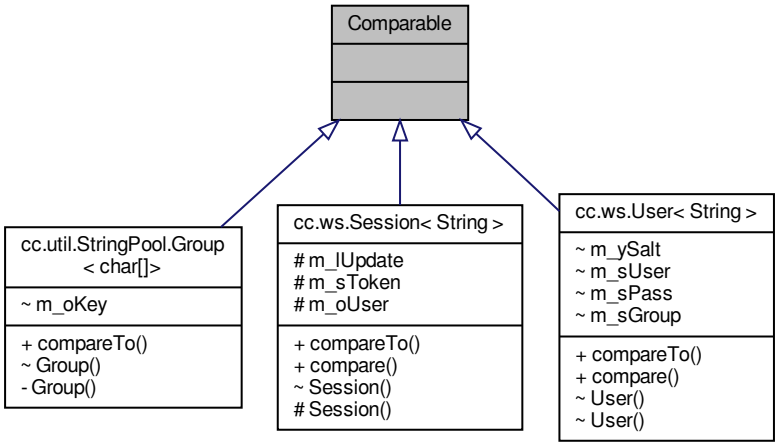
Referenced by cc.util.BufferedInStream.BufferedInStream(), and cc.util.BufferedInStream.read().

The documentation for this class was generated from the following file:

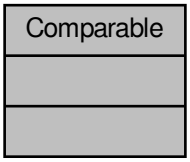
- src/cc/util/BufferedInStream.java

7.3 Comparable Class Reference

Inheritance diagram for Comparable:



Collaboration diagram for Comparable:

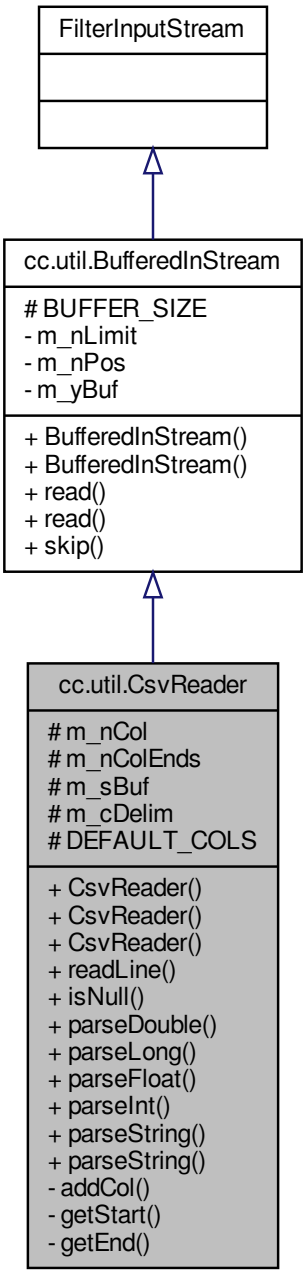


The documentation for this class was generated from the following file:

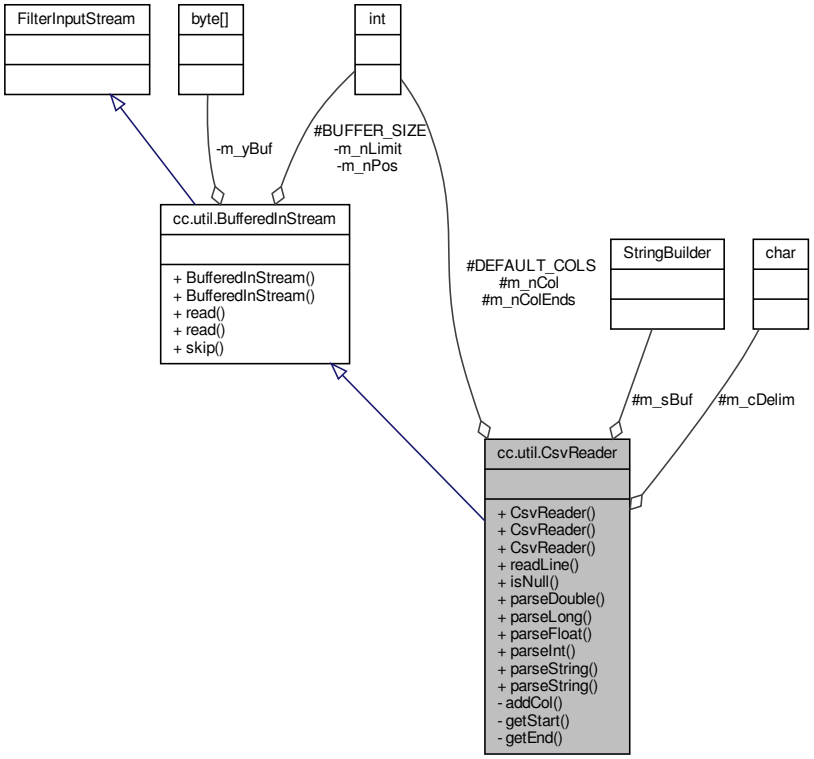
- src/cc/util/StringPool.java

7.4 cc.util.CsvReader Class Reference

Inheritance diagram for cc.util.CsvReader:



Collaboration diagram for cc.util.CsvReader:



Public Member Functions

- CsvReader (InputStream oInputStream, int nCols)
- CsvReader (InputStream oInputStream)
- CsvReader (InputStream oInputStream, char cDelim)
- int readLine () throws IOException
- boolean isNull (int nCol) throws IndexOutOfBoundsException
- double parseDouble (int nCol) throws IndexOutOfBoundsException, NumberFormatException
- long parseLong (int nCol) throws IndexOutOfBoundsException, NumberFormatException
- float parseFloat (int nCol) throws IndexOutOfBoundsException, NumberFormatException
- int parseInt (int nCol) throws IndexOutOfBoundsException, NumberFormatException
- String parseString (int nCol) throws IndexOutOfBoundsException
- int parseString (StringBuilder sBuf, int nCol) throws IndexOutOfBoundsException, NullPointerException

Protected Attributes

- int m_nCol
- int[ ] m_nColEnds
- StringBuilder m_sBuf = new StringBuilder(BUFFER_SIZE)
- char m_cDelim = ','

Static Protected Attributes

- static final int DEFAULT_COLS = 80

Private Member Functions

- void addCol ()
- int getStart (int nCol)
- int getEnd (int nCol)

7.4.1 Detailed Description

Definition at line 7 of file CsvReader.java.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 CsvReader() [1/3]

```
cc.util.CsvReader.CsvReader (
            InputStream oInputStream,
            int nCols )  [inline]
```

Definition at line 16 of file CsvReader.java.

```
00017     {
00018         super(oInputStream);
00019         m_nColEnds = new int[nCols];
00020     }
```

References cc.util.CsvReader.m_nColEnds.

7.4.2.2 CsvReader() [2/3]

```
cc.util.CsvReader.CsvReader (
            InputStream oInputStream )  [inline]
```

Definition at line 23 of file CsvReader.java.

```
00024     {
00025         this(oInputStream, DEFAULT_COLS);
00026     }
```

References cc.util.CsvReader.DEFAULT_COLS.

7.4.2.3 CsvReader() [3/3]

```
cc.util.CsvReader.CsvReader (
            InputStream oInputStream,
            char cDelim )  [inline]
```

Definition at line 29 of file CsvReader.java.

```
00030     {
00031         this(oInputStream, DEFAULT_COLS);
00032         m_cDelim = cDelim;
00033     }
```

References cc.util.CsvReader.DEFAULT_COLS, and cc.util.CsvReader.m_cDelim.

7.4.3 Member Function Documentation

7.4.3.1 addCol()

```
void cc.util.CsvReader.addCol ( )  [inline], [private]
```

Definition at line 63 of file CsvReader.java.

```
00064     {
00065         if (m_nCol == m_nColEnds.length) // extend column end array
00066         {
00067             int[] nColEnds = new int[m_nCol * 2];
00068             System.arraycopy(m_nColEnds, 0, nColEnds, 0, m_nCol);
00069             m_nColEnds = nColEnds;
00070         }
00071         m_nColEnds[m_nCol++] = m_sBuf.length();
00072     }
```

References cc.util.CsvReader.m_nCol, cc.util.CsvReader.m_nColEnds, and cc.util.CsvReader.m_sBuf.

Referenced by cc.util.CsvReader.readLine().

Here is the caller graph for this function:

7.4.3.2 getEnd()

```
int cc.util.CsvReader.getEnd (
             int nCol )  [inline], [private]
```

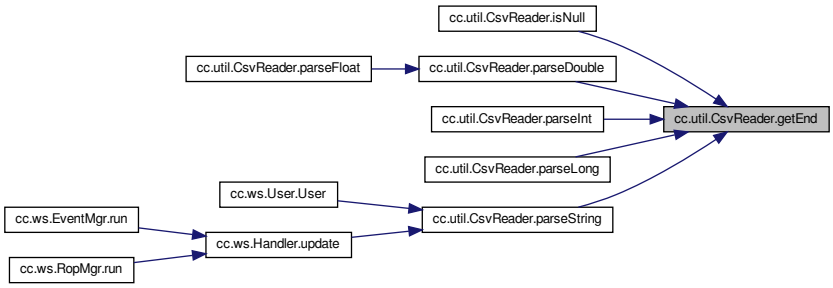
Definition at line 88 of file CsvReader.java.

```
00089     {
00090         if (nCol < m_nCol)
00091             return m_nColEnds[nCol];
00092 
00093         return -1; // force index out of bounds
00094     }
```

References cc.util.CsvReader.m_nCol, and cc.util.CsvReader.m_nColEnds.

Referenced by cc.util.CsvReader.isNull(), cc.util.CsvReader.parseDouble(), cc.util.CsvReader.parseInt(), cc.util.CsvReader.parseLong(), and cc.util.CsvReader.parseString().

Here is the caller graph for this function:



7.4.3.3 getStart()

```
int cc.util.CsvReader.getStart (
            int nCol )  [inline], [private]
```

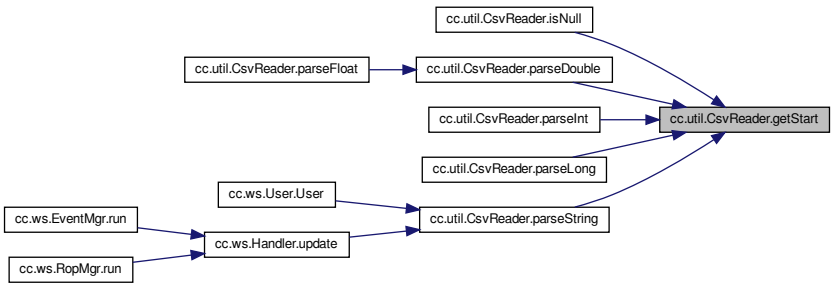
Definition at line 75 of file CsvReader.java.

```
00076     {
00077         if (nCol < m_nCol)
00078         {
00079             if (nCol == 0)
00080                 return 0;
00081 
00082             return m_nColEnds[nCol - 1];
00083         }
00084         return -1; // force index out of bounds
00085     }
```

References cc.util.CsvReader.m_nCol, and cc.util.CsvReader.m_nColEnds.

Referenced by cc.util.CsvReader.isNull(), cc.util.CsvReader.parseDouble(), cc.util.CsvReader.parseInt(), cc.util.CsvReader.parseLong(), and cc.util.CsvReader.parseString().

Here is the caller graph for this function:



7.4.3.4 isNull()

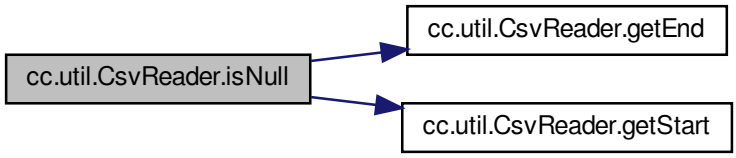
```
boolean cc.util.CsvReader.isNull (
            int nCol ) throws IndexOutOfBoundsException  [inline]
```

Definition at line 97 of file CsvReader.java.

```
00099     {
00100         return (getEnd(nCol) - getStart(nCol) == 0);
00101     }
```

References cc.util.CsvReader.getEnd(), and cc.util.CsvReader.getStart().

Here is the call graph for this function:



7.4.3.5 parseDouble()

```
double cc.util.CsvReader.parseDouble (
            int nCol ) throws IndexOutOfBoundsException, NumberFormatException  [inline]
```

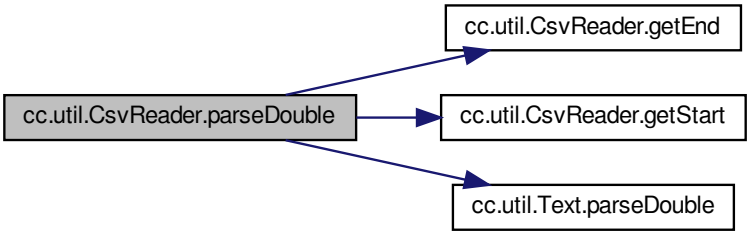
Definition at line 104 of file CsvReader.java.

```
00106     {
00107         return Text.parseDouble(m_sBuf, getStart(nCol), getEnd(nCol));
00108     }
```

References cc.util.CsvReader.getEnd(), cc.util.CsvReader.getStart(), cc.util.CsvReader.m_sBuf, and cc.util.Text.parseDouble().

Referenced by cc.util.CsvReader.parseFloat().

Here is the call graph for this function:



Here is the caller graph for this function:

7.4.3.6 parseFloat()

```
float cc.util.CsvReader.parseFloat (
            int nCol ) throws IndexOutOfBoundsException, NumberFormatException  [inline]
```

Definition at line 118 of file CsvReader.java.

```
00120     {
00121         return (float)parseDouble(nCol);
00122     }
```

References cc.util.CsvReader.parseDouble().

Here is the call graph for this function:

7.4.3.7 parseInt()

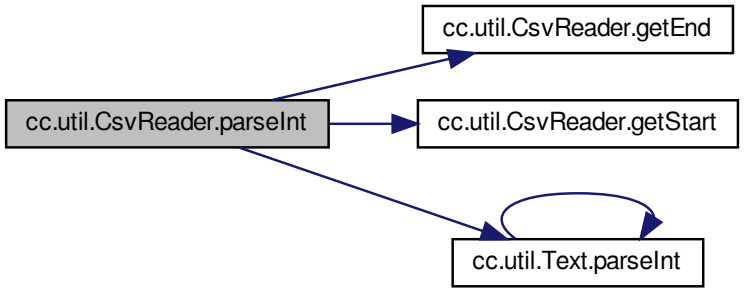
```
int cc.util.CsvReader.parseInt (
            int nCol ) throws IndexOutOfBoundsException, NumberFormatException  [inline]
```

Definition at line 125 of file CsvReader.java.

```
00127     {
00128         return Text.parseInt(m_sBuf, getStart(nCol), getEnd(nCol));
00129     }
```

References cc.util.CsvReader.getEnd(), cc.util.CsvReader.getStart(), cc.util.CsvReader.m_sBuf, and cc.util.Text.parseInt().

Here is the call graph for this function:



7.4.3.8 parseLong()

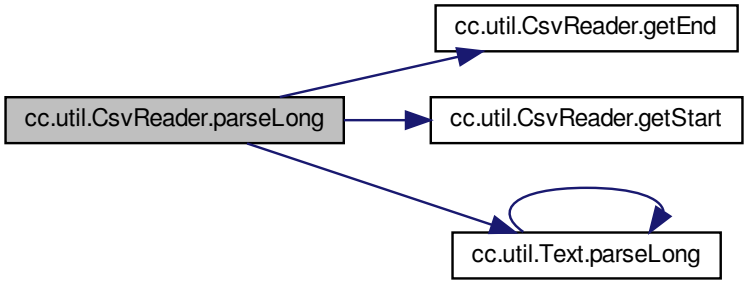
```
long cc.util.CsvReader.parseLong (
            int nCol ) throws IndexOutOfBoundsException, NumberFormatException  [inline]
```

Definition at line 111 of file CsvReader.java.

```
00113     {
00114         return Text.parseLong(m_sBuf, getStart(nCol), getEnd(nCol));
00115     }
```

References cc.util.CsvReader.getEnd(), cc.util.CsvReader.getStart(), cc.util.CsvReader.m_sBuf, and cc.util.Text.parseLong().

Here is the call graph for this function:



7.4.3.9 parseString() [1/2]

```
String cc.util.CsvReader.parseString (
            int nCol ) throws IndexOutOfBoundsException  [inline]
```

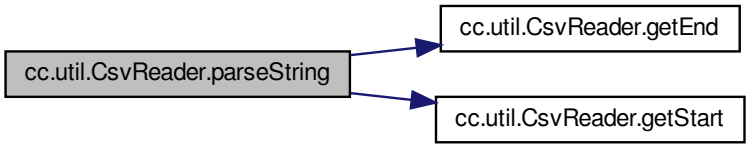
Definition at line 132 of file CsvReader.java.

```
00134     {
00135         return m_sBuf.substring(getStart(nCol), getEnd(nCol));
00136     }
```

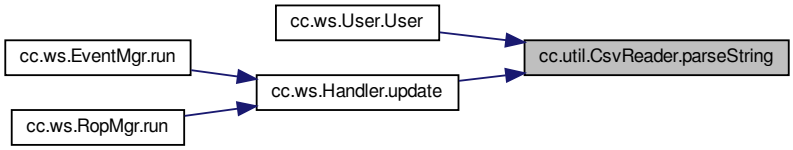
References cc.util.CsvReader.getEnd(), cc.util.CsvReader.getStart(), and cc.util.CsvReader.m_sBuf.

Referenced by cc.ws.User.User(), and cc.ws.Handler.update().

Here is the call graph for this function:



Here is the caller graph for this function:



7.4.3.10 parseString() [2/2]

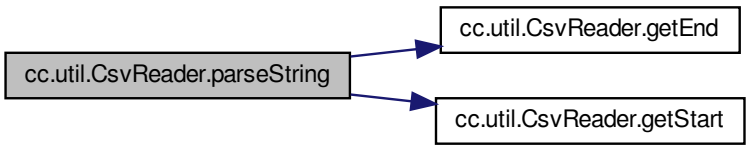
```
int cc.util.CsvReader.parseString (
            StringBuilder sBuf,
            int nCol ) throws IndexOutOfBoundsException, NullPointerException  [inline]
```

Definition at line 139 of file CsvReader.java.

```
00141     {
00142         sBuf.setLength(0); // clear provided buffer
00143         sBuf.append(m_sBuf, getStart(nCol), getEnd(nCol));
00144         return sBuf.length();
00145     }
```

References cc.util.CsvReader.getEnd(), cc.util.CsvReader.getStart(), and cc.util.CsvReader.m_sBuf.

Here is the call graph for this function:



7.4.3.11 readLine()

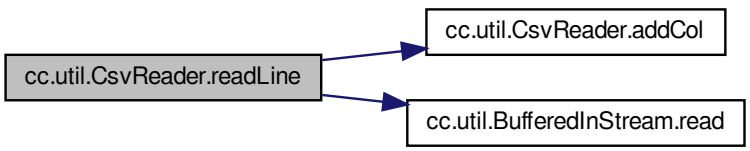
```
int cc.util.CsvReader.readLine ( ) throws IOException  [inline]
```

Definition at line 36 of file CsvReader.java.

```
00038     {
00039         m_nCol = 0; // reset column index
00040         m_sBuf.setLength(0); // reset line buffer
00041 
00042         boolean bGo = true;
00043         int nChar;
00044         while (bGo && (nChar = read()) >= 0) // don't advance on line complete
00045         {
00046             if (nChar == m_cDelim || nChar < ' ')
00047             {
00048                 bGo = (nChar != '\n');
00049                 if (nChar != '\r') // ignore carriage return
00050                     addCol(); // column found
00051             }
00052             else
00053                 m_sBuf.append((char)nChar);
00054         }
00055 
00056         if (bGo && m_nCol > 0) // check for missing final newline
00057             addCol();
00058 
00059         return m_nCol; // discovered column count
00060     }
```

References cc.util.CsvReader.addCol(), cc.util.CsvReader.m_cDelim, cc.util.CsvReader.m_nCol, cc.util.CsvReader.m_sBuf, and cc.util.BufferedInStream.read().

Here is the call graph for this function:



7.4.4 Member Data Documentation

7.4.4.1 DEFAULT_COLS

```
final int cc.util.CsvReader.DEFAULT_COLS = 80  [static], [protected]
```

Definition at line 9 of file CsvReader.java.

Referenced by cc.util.CsvReader.CsvReader().

7.4.4.2 m_cDelim

```
char cc.util.CsvReader.m_cDelim = ','  [protected]
```

Definition at line 14 of file CsvReader.java.

Referenced by cc.util.CsvReader.CsvReader(), and cc.util.CsvReader.readLine().

7.4.4.3 m_nCol

```
int cc.util.CsvReader.m_nCol  [protected]
```

Definition at line 11 of file CsvReader.java.

Referenced by cc.util.CsvReader.addCol(), cc.util.CsvReader.getEnd(), cc.util.CsvReader.getStart(), and cc.util.CsvReader.readLine().

7.4.4.4 m_nColEnds

```
int [] cc.util.CsvReader.m_nColEnds  [protected]
```

Definition at line 12 of file CsvReader.java.

Referenced by cc.util.CsvReader.CsvReader(), cc.util.CsvReader.addCol(), cc.util.CsvReader.getEnd(), and cc.util.CsvReader.getStart().

7.4.4.5 m_sBuf

```
StringBuilder cc.util.CsvReader.m_sBuf = new StringBuilder(BUFFER_SIZE)  [protected]
```

Definition at line 13 of file CsvReader.java.

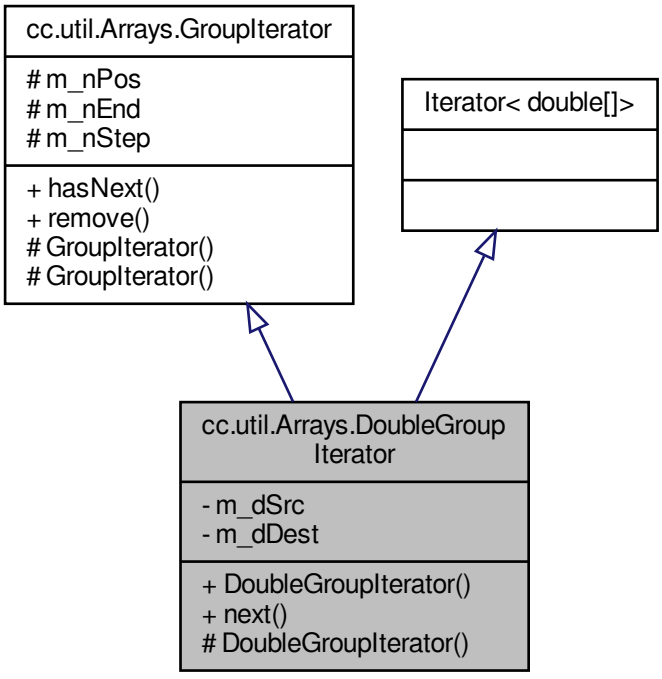
Referenced by cc.util.CsvReader.addCol(), cc.util.CsvReader.parseDouble(), cc.util.CsvReader.parseInt(), cc.util.CsvReader.parseLong(), cc.util.CsvReader.parseString(), and cc.util.CsvReader.readLine().

The documentation for this class was generated from the following file:

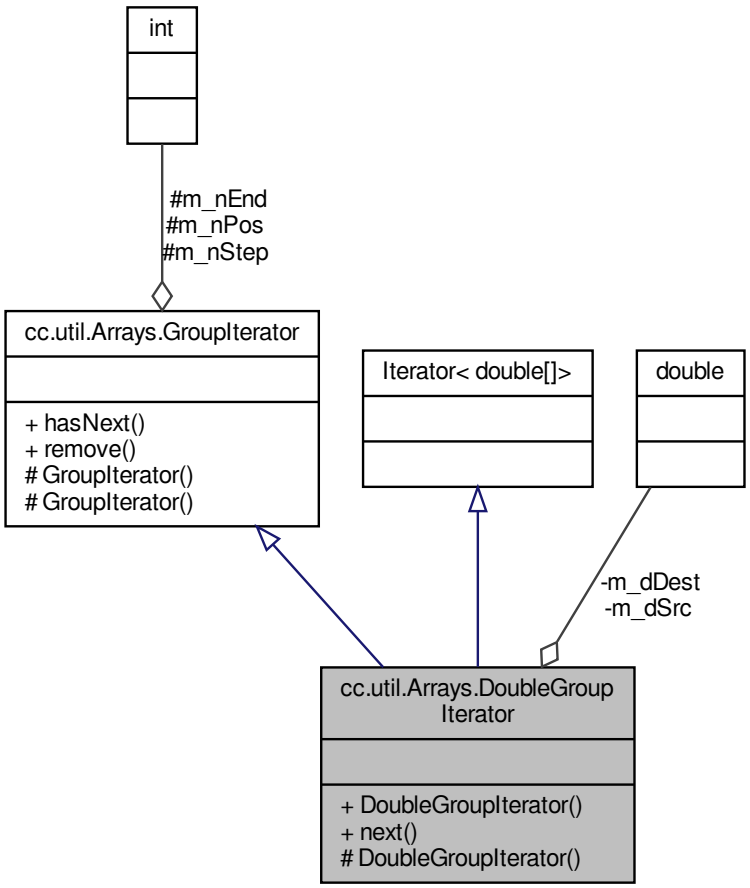
- src/cc/util/CsvReader.java

7.5 cc.util.Arrays.DoubleGroupIterator Class Reference

Inheritance diagram for cc.util.Arrays.DoubleGroupIterator:



Collaboration diagram for cc.util.Arrays.DoubleGroupIterator:



Public Member Functions

- DoubleGroupIterator (double[ ] dSrc, double[ ] dDest, int nStart, int nStep) throws IllegalArgumentException
- double[ ] next ()

Protected Member Functions

- DoubleGroupIterator ()

Private Attributes

- double[ ] m_dSrc
- double[ ] m_dDest

Additional Inherited Members

7.5.1 Detailed Description

Definition at line 226 of file Arrays.java.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 DoubleGroupIterator() [1/2]

```
cc.util.Arrays.DoubleGroupIterator.DoubleGroupIterator ( )  [inline], [protected]
```

Definition at line 232 of file Arrays.java.

```
00233         {
00234         }
```

7.5.2.2 DoubleGroupIterator() [2/2]

```
cc.util.Arrays.DoubleGroupIterator.DoubleGroupIterator (
              double[] dSrc,
              double[] dDest,
              int nStart,
              int nStep ) throws IllegalArgumentException  [inline]
```

Definition at line 237 of file Arrays.java.

```
00239         {
00240             super(nStart, (int)dSrc[0], dDest.length, nStep);
00241             if (dSrc.length == 0 || dDest.length == 0 || dSrc.length < dDest.length + 1 || nStart < 0
       || nStep <= 0)
00242                 throw new IllegalArgumentException();
00243             m_dDest = dDest;
00244             m_dSrc = dSrc; // local reference to values
00245         }
```

References cc.util.Arrays.DoubleGroupIterator.m_dDest, and cc.util.Arrays.DoubleGroupIterator.m_dSrc.

7.5.3 Member Function Documentation

7.5.3.1 next()

```
double[] cc.util.Arrays.DoubleGroupIterator.next ( )  [inline]
```

Definition at line 249 of file Arrays.java.

```
00250         {
00251             System.arraycopy(m_dSrc, m_nPos, m_dDest, 0, m_dDest.length);
00252             m_nPos += m_nStep; // shift to next group position
00253             return m_dDest;
00254         }
```

References cc.util.Arrays.DoubleGroupIterator.m_dDest, cc.util.Arrays.DoubleGroupIterator.m_dSrc, cc.util.Arrays.GroupIterator.m_nI
and cc.util.Arrays.GroupIterator.m_nStep.

7.5.4 Member Data Documentation

7.5.4.1 m_dDest

```
double [] cc.util.Arrays.DoubleGroupIterator.m_dDest  [private]
```

Definition at line 229 of file Arrays.java.

Referenced by cc.util.Arrays.DoubleGroupIterator.DoubleGroupIterator(), and cc.util.Arrays.DoubleGroupIterator.next().

7.5.4.2 m_dSrc

```
double [] cc.util.Arrays.DoubleGroupIterator.m_dSrc  [private]
```

Definition at line 228 of file Arrays.java.

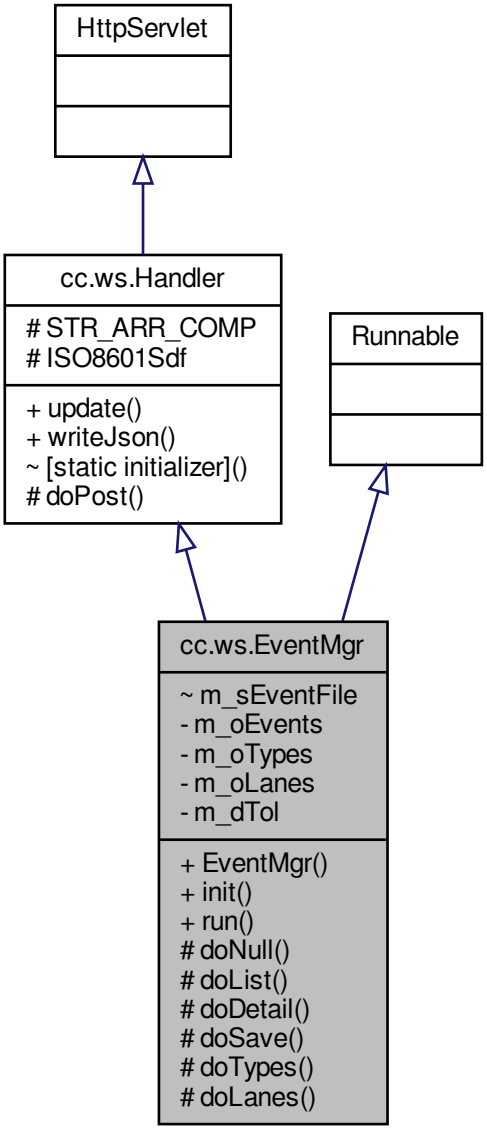
Referenced by cc.util.Arrays.DoubleGroupIterator.DoubleGroupIterator(), and cc.util.Arrays.DoubleGroupIterator.next().

The documentation for this class was generated from the following file:

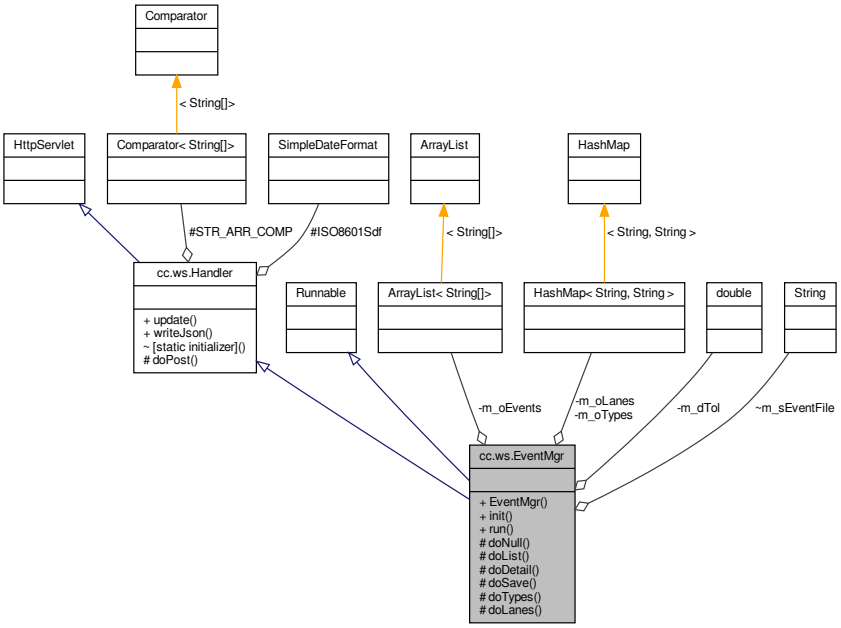
- src/cc/util/Arrays.java

7.6 cc.ws.EventMgr Class Reference

Inheritance diagram for cc.ws.EventMgr:



Collaboration diagram for cc.ws.EventMgr:



Public Member Functions

- EventMgr ()
- void init ()
- void run ()

Protected Member Functions

- void doNull (Session oSess, HttpServletRequest oReq, PrintWriter oOut)
- void doList (Session oSess, HttpServletRequest oReq, PrintWriter oOut)
- void doDetail (Session oSess, HttpServletRequest oReq, PrintWriter oOut)
- void doSave (Session oSess, HttpServletRequest oReq, PrintWriter oOut) throws IOException
- void doTypes (Session oSess, HttpServletRequest oReq, PrintWriter oOut)
- void doLanes (Session oSess, HttpServletRequest oReq, PrintWriter oOut)

Package Attributes

- String m_sEventFile

Private Attributes

- final ArrayList< String[ ]> m_oEvents = new ArrayList()
- final HashMap< String, String > m_oTypes = new HashMap()
- final HashMap< String, String > m_oLanes = new HashMap()
- double m_dTol = 0.00001

Additional Inherited Members

7.6.1 Detailed Description

Definition at line 18 of file EventMgr.java.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 EventMgr()

```
cc.ws.EventMgr.EventMgr ( )  [inline]
```

Definition at line 28 of file EventMgr.java.

```
00029     {
00030         m_oTypes.put("1", "work zone");
00031         m_oTypes.put("2", "incident");
00032 
00033         m_oLanes.put("8193", "all");
00034         m_oLanes.put("8195", "left-lane");
00035         m_oLanes.put("8196", "right-lane");
00036         m_oLanes.put("81952", "left-2-lanes");
00037         m_oLanes.put("81953", "left-3-lanes");
00038         m_oLanes.put("81962", "right-2-lanes");
00039         m_oLanes.put("81963", "right-3-lanes");
00040         m_oLanes.put("8197", "center");
00041         m_oLanes.put("8198", "middle-lane");
00042         m_oLanes.put("8200", "right-turning-lane");
00043         m_oLanes.put("8201", "left-turning-lane");
00044         m_oLanes.put("8239", "right-exit-lane");
00045         m_oLanes.put("8240", "left-exit-lane");
00046         m_oLanes.put("8241", "right-merging-lane");
00047         m_oLanes.put("8242", "left-merging-lane");
00048         m_oLanes.put("8202", "right-exit-ramp");
00049         m_oLanes.put("8243", "right-second-exit-ramp");
00050         m_oLanes.put("8203", "right-entrance-ramp");
00051         m_oLanes.put("8245", "right-second-entrance-ramp");
00052         m_oLanes.put("8204", "left-exit-ramp");
00053         m_oLanes.put("8244", "left-second-exit-ramp");
00054         m_oLanes.put("8205", "left-entrance-ramp");
00055         m_oLanes.put("8246", "left-second-entrance-ramp");
00056         m_oLanes.put("82281", "sidewalk");
00057         m_oLanes.put("8228", "bike-lane");
00058         m_oLanes.put("0", "none");
00059         m_oLanes.put("8247", "unknown");
00060         m_oLanes.put("81980", "alternate-flow-lane");
00061         m_oLanes.put("81951", "shift-left");
00062         m_oLanes.put("81961", "shift-right");
00063     }
```

References cc.ws.EventMgr.m_oLanes, and cc.ws.EventMgr.m_oTypes.

7.6.3 Member Function Documentation

7.6.3.1 doDetail()

```
void cc.ws.EventMgr.doDetail (
            Session oSess,
            HttpServletRequest oReq,
            PrintWriter oOut )  [inline], [protected]
```

Definition at line 247 of file EventMgr.java.

```
00248     {
00249         String sEventId = oReq.getParameter("id");
00250         if (sEventId == null || m_oEvents.isEmpty())
00251             return;
00252 
00253         synchronized(m_oEvents) // ensure any changes are committed
00254         {
00255             String[] sSearch = new String[]{sEventId};
00256             int nIndex = Collections.binarySearch(m_oEvents, sSearch, STR_ARR_COMP);
00257             if (nIndex < 0) // list only contains most recent updates
00258             {
00259                 oOut.write("{}");
00260                 return;
00261             }
00262 
00263             String[] sEvent = m_oEvents.get(nIndex);
00264             oOut.write("{");
00265             writeJson(oOut, sEvent[0], sEvent[sEvent.length - 1]);
00266             oOut.write("}");
00267         }
00268     }
```

References cc.ws.EventMgr.m_oEvents, cc.ws.Handler.STR_ARR_COMP, and cc.ws.Handler.writeJson().

Here is the call graph for this function:

7.6.3.2 doLanes()

```
void cc.ws.EventMgr.doLanes (
            Session oSess,
            HttpServletRequest oReq,
            PrintWriter oOut )  [inline], [protected]
```

Definition at line 326 of file EventMgr.java.

```
00327     {
00328         oOut.write("{");
00329         if (m_oLanes.size() > 0)
00330         {
00331             StringBuilder sBuffer = new StringBuilder();
00332             m_oLanes.entrySet().forEach(oEntry -> sBuffer.append(String.format("\"%s\":\"%s\",",
      oEntry.getKey(), oEntry.getValue())));
00333             sBuffer.setLength(sBuffer.length() - 1); // remove last comma
00334             oOut.write(sBuffer.toString());
00335         }
00336         oOut.write("}");
00337     }
```

References cc.ws.EventMgr.m_oLanes.

7.6.3.3 doList()

```
void cc.ws.EventMgr.doList (
            Session oSess,
            HttpServletRequest oReq,
            PrintWriter oOut )  [inline], [protected]
```

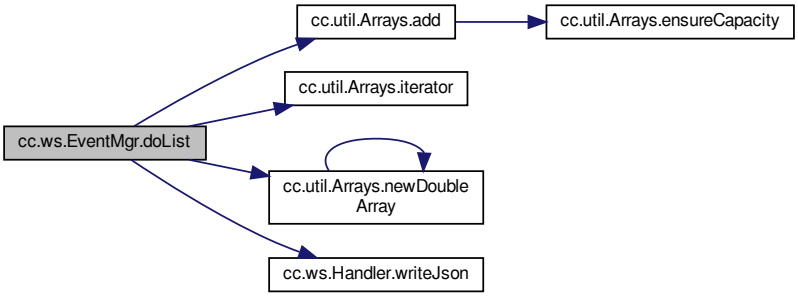
Definition at line 147 of file EventMgr.java.

```
00148     {
00149         double dLat1;
00150         double dLat2;
00151         double dLon1;
00152         double dLon2;
00153         try
00154         {
00155             dLat1 = Double.parseDouble(oReq.getParameter("lat1")); // filter by selected point
00156             dLat2 = Double.parseDouble(oReq.getParameter("lat2"));
00157             if (dLat1 > dLat2)
00158             {
00159                 double dTemp = dLat2;
00160                 dLat2 = dLat1;
00161                 dLat1 = dTemp;
00162             }
00163 
00164             dLon1 = Double.parseDouble(oReq.getParameter("lon1"));
00165             dLon2 = Double.parseDouble(oReq.getParameter("lon2"));
00166             if (dLon1 > dLon2)
00167             {
00168                 double dTemp = dLon2;
00169                 dLon2 = dLon1;
00170                 dLon1 = dTemp;
00171             }
00172         }
00173         catch (Exception oEx)
00174         {
00175             oOut.write("{}"); // send empty object if error getting parameters
00176             return;
00177         }
00178 
00179         oOut.write("{");
00180         synchronized(m_oEvents)
00181         {
00182             double[] dPoints = Arrays.newDoubleArray();
00183             double[] dSeg = new double[4];
00184             int nCount = 0;
00185             for (String[] sEvent : m_oEvents)
00186             {
00187                 dPoints[0] = 1;
00188                 String sData = sEvent[sEvent.length - 1];
00189                 int nStart = sData.indexOf("\"pts\":[");
00190                 if (nStart < 0)
00191                     continue;
00192 
00193                 nStart += "\"pts\":[".length();
00194                 int nEnd = sData.indexOf("]", nStart);
00195                 String[] sOrdinates = sData.substring(nStart, nEnd).split(",");
00196                 for (int i = 0; i < sOrdinates.length;)
00197                 {
00198                     dPoints = Arrays.add(dPoints,
00199                        Math.round(Double.parseDouble(sOrdinates[i++]) * 10000000.0 + 0.000001) /
      10000000.0,
00200                        Math.round(Double.parseDouble(sOrdinates[i++]) * 10000000.0 + 0.000001) /
      10000000.0);
00201                 }
00202 
00203                 Iterator<double[]> oIt = Arrays.iterator(dPoints, dSeg, 1, 2);
00204                 while (oIt.hasNext())
00205                 {
00206                     oIt.next();
00207                     double dSegLatMin = dSeg[0];
00208                     double dSegLatMax = dSeg[2];
00209                     double dSegLonMin = dSeg[1];
00210                     double dSegLonMax = dSeg[3];
00211 
00212                     if (dSegLatMin > dSegLatMax)
00213                     {
00214                         double dTemp = dSegLatMax;
00215                         dSegLatMax = dSegLatMin;
00216                         dSegLatMin = dTemp;
00217                     }
00218 
00219                     if (dSegLonMin > dSegLonMax)
00220                     {
```

```
00221                           double dTemp = dSegLonMax;
00222                           dSegLonMax = dSegLonMin;
00223                           dSegLonMin = dTemp;
00224                       }
00225
00226                       dSegLatMin -= m_dTol;
00227                       dSegLatMax += m_dTol;
00228                       dSegLonMin -= m_dTol;
00229                       dSegLonMax += m_dTol;
00230
00231                       if (dLon2 < dSegLonMin || dLon1 > dSegLonMax ||
00232                           dLat2 < dSegLatMin || dLat1 > dSegLatMax)
00233                           continue;
00234
00235                       if (nCount++ > 0)
00236                           oOut.write(',');
00237                       writeJson(oOut, sEvent[0], sData);
00238
00239                       break;
00240                   }
00241               }
00242           }
00243           oOut.write("}");
00244       }
```

References cc.util.Arrays.add(), cc.util.Arrays.iterator(), cc.ws.EventMgr.m_dTol, cc.ws.EventMgr.m_oEvents, cc.util.Arrays.newDoubleArray(), and cc.ws.Handler.writeJson().

Here is the call graph for this function:



7.6.3.4 doNull()

```
void cc.ws.EventMgr.doNull (
            Session oSess,
            HttpServletRequest oReq,
            PrintWriter oOut )  [inline], [protected]
```

Definition at line 109 of file EventMgr.java.

```
00110     {
00111         oOut.write("{");
00112         synchronized(m_oEvents)
00113         {
00114             int nCount = 0;
00115             for (String[] sEvent : m_oEvents)
00116             {
00117                 String sData = sEvent[sEvent.length - 1];
00118                 int nStart = sData.indexOf("\"pts\":[");
00119                 if (nStart < 0)
00120                     continue;
00121
```

```
00122                 nStart += "\"pts\":[".length();
00123                 int nEnd = sData.indexOf("]", nStart);
00124                 String[] sOrdinates = sData.substring(nStart, nEnd).split(",");
00125                 if (nCount++ > 0)
00126                     oOut.write(",");
00127 
00128                 oOut.write(String.format("\"%s\":[", sEvent[0]));
00129 
00130                 int nSize = sOrdinates.length;
00131                 if (nSize > 0)
00132                 {
00133                     oOut.write(sOrdinates[0]);
00134                     for (int i = 1; i < nSize; i++)
00135                     {
00136                         oOut.write(",");
00137                         oOut.write(sOrdinates[i]); // {"uuid" : pt array, "uuid" : pt array, ...}
00138                     }
00139                 }
00140                 oOut.write("]");
00141             }
00142         }
00143         oOut.write("}");
00144     }
```

References cc.ws.EventMgr.m_oEvents.

7.6.3.5 doSave()

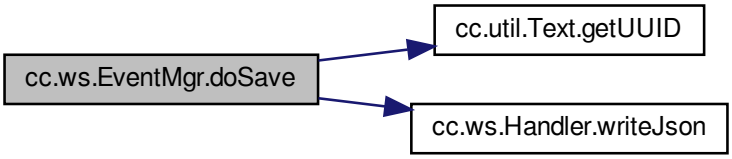
```
void cc.ws.EventMgr.doSave (
            Session oSess,
            HttpServletRequest oReq,
            PrintWriter oOut ) throws IOException  [inline], [protected]
```

Definition at line 271 of file EventMgr.java.

```
00273     {
00274         String sEventId = oReq.getParameter("id");
00275         if (sEventId == null || sEventId.length() == 0) // new event condition
00276             sEventId = Text.getUUID();
00277 
00278         synchronized(m_oEvents)
00279         {
00280             String sUsername = oSess.m_oUser.m_sUser; // set event parameters
00281 
00282             String[] sSearch = new String[]{sEventId};
00283             int nIndex = Collections.binarySearch(m_oEvents, sSearch, STR_ARR_COMP);
00284             if (nIndex < 0)
00285             {
00286                 nIndex = ~nIndex;
00287                 m_oEvents.add(nIndex, new String[]{sEventId, sUsername, null,
      oReq.getParameter("data")}); // insert new event
00288             }
00289             String[] sEvent = m_oEvents.get(nIndex);
00290             synchronized (ISO8601Sdf)
00291             {
00292                 sEvent[2] = ISO8601Sdf.format(System.currentTimeMillis()); // set update time last
00293             }
00294 
00295             try (BufferedWriter oFileOut = new BufferedWriter(new FileWriter(m_sEventFile, true)))
00296             {
00297                 oFileOut.write(sEvent[0]);
00298                 for (int i = 1; i < sEvent.length; i++)
00299                 {
00300                     oFileOut.write("\t");
00301                     oFileOut.write(sEvent[i]);
00302                 }
00303                 oFileOut.write("\n");
00304             }
00305             oOut.write("{");
00306             writeJson(oOut, sEvent[0], sEvent[sEvent.length - 1]); // id:data
00307             oOut.write("}");
00308         }
00309     }
```

References cc.util.Text.getUUID(), cc.ws.Handler.ISO8601Sdf, cc.ws.EventMgr.m_oEvents, cc.ws.EventMgr.m_sEventFile, cc.ws.Handler.STR_ARR_COMP, and cc.ws.Handler.writeJson().

Here is the call graph for this function:



7.6.3.6 doTypes()

```
void cc.ws.EventMgr.doTypes (
            Session oSess,
            HttpServletRequest oReq,
            PrintWriter oOut )  [inline], [protected]
```

Definition at line 312 of file EventMgr.java.

```
00313     {
00314         oOut.write("{");
00315         if (m_oTypes.size() > 0)
00316         {
00317             StringBuilder sBuffer = new StringBuilder();
00318             m_oTypes.entrySet().forEach(oEntry -> sBuffer.append(String.format("\"%s\":\"%s\",",
      oEntry.getKey(), oEntry.getValue())));
00319             sBuffer.setLength(sBuffer.length() - 1); // remove last comma
00320             oOut.write(sBuffer.toString());
00321         }
00322         oOut.write("}");
00323     }
```

References cc.ws.EventMgr.m_oTypes.

7.6.3.7 init()

```
void cc.ws.EventMgr.init ( )  [inline]
```

Definition at line 67 of file EventMgr.java.

```
00068     {
00069         String sEventFile = getServletConfig().getInitParameter("eventfile");
00070         if (sEventFile != null && sEventFile.length() > 0)
00071         {
00072             m_sEventFile = sEventFile;
00073             new Thread(this).start();
00074         }
00075         String sTol = getServletConfig().getInitParameter("tol");
00076         if (sTol != null && sTol.length() > 0)
00077             m_dTol = Double.parseDouble(sTol);
00078     }
```

References cc.ws.EventMgr.m_dTol, and cc.ws.EventMgr.m_sEventFile.

7.6.3.8 run()

```
void cc.ws.EventMgr.run ( )  [inline]
```

Definition at line 82 of file EventMgr.java.

```
00083     {
00084         try (CsvReader oIn = new CsvReader(new FileInputStream(m_sEventFile), '\t'))
00085         {
00086             int nCols;
00087             String[] sSearch = new String[1];
00088             synchronized(m_oEvents)
00089             {
00090                 while ((nCols = oIn.readLine()) > 0)
00091                 {
00092                     sSearch[0] = oIn.parseString(0);
00093                     int nIndex = Collections.binarySearch(m_oEvents, sSearch, STR_ARR_COMP); // search
    for id in list
00094                     if (nIndex < 0)
00095                     {
00096                         nIndex = ~nIndex;
00097                         m_oEvents.add(nIndex, new String[nCols]); // right now we have
    uuid,user,timestamp,status,JSON
00098                     }
00099                     update(m_oEvents.get(nIndex), nCols, oIn); // replace with most recent update
00100                 }
00101             }
00102         }
00103         catch (Exception oEx)
00104         {
00105         }
00106     }
```

References cc.ws.EventMgr.m_oEvents, cc.ws.EventMgr.m_sEventFile, cc.ws.Handler.STR_ARR_COMP, and cc.ws.Handler.update().

Here is the call graph for this function:

7.6.4 Member Data Documentation

7.6.4.1 m_dTol

```
double cc.ws.EventMgr.m_dTol = 0.00001  [private]
```

Definition at line 23 of file EventMgr.java.

Referenced by cc.ws.EventMgr.doList(), and cc.ws.EventMgr.init().

7.6.4.2 m_oEvents

```
final ArrayList<String[]> cc.ws.EventMgr.m_oEvents = new ArrayList()  [private]
```

Definition at line 20 of file EventMgr.java.

Referenced by cc.ws.EventMgr.doDetail(), cc.ws.EventMgr.doList(), cc.ws.EventMgr.doNull(), cc.ws.EventMgr.doSave(), and cc.ws.EventMgr.run().

7.6.4.3 m_oLanes

```
final HashMap<String, String> cc.ws.EventMgr.m_oLanes = new HashMap()  [private]
```

Definition at line 22 of file EventMgr.java.

Referenced by cc.ws.EventMgr.EventMgr(), and cc.ws.EventMgr.doLanes().

7.6.4.4 m_oTypes

```
final HashMap<String, String> cc.ws.EventMgr.m_oTypes = new HashMap()  [private]
```

Definition at line 21 of file EventMgr.java.

Referenced by cc.ws.EventMgr.EventMgr(), and cc.ws.EventMgr.doTypes().

7.6.4.5 m_sEventFile

```
String cc.ws.EventMgr.m_sEventFile  [package]
```

Definition at line 25 of file EventMgr.java.

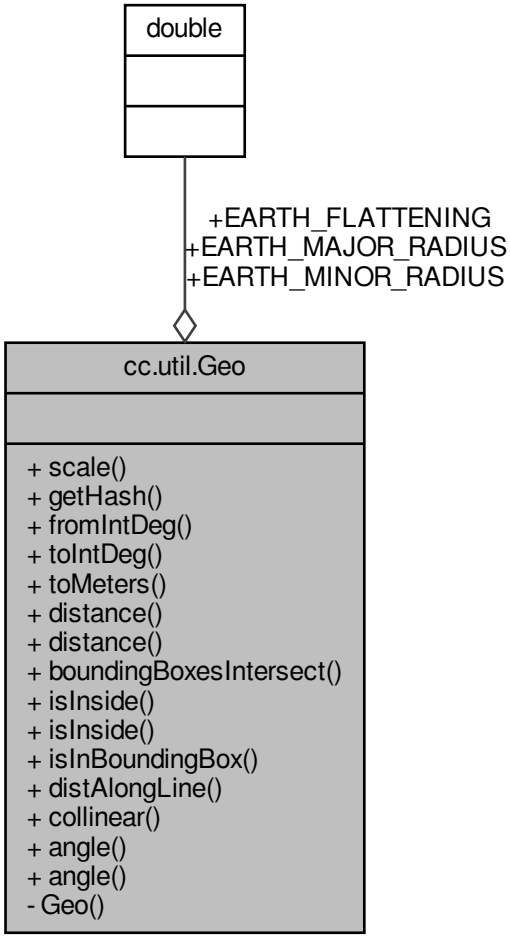
Referenced by cc.ws.EventMgr.doSave(), cc.ws.EventMgr.init(), and cc.ws.EventMgr.run().

The documentation for this class was generated from the following file:

- src/cc/ws/EventMgr.java

7.7 cc.util.Geo Class Reference

Collaboration diagram for cc.util.Geo:



Static Public Member Functions

- static int scale (int nVal)
- static int getHash (int nLat, int nLon)
- static double fromIntDeg (int nOrd)
- static int toIntDeg (double dOrd)
- static double toMeters (int nOrd)
- static double distance (int nXi, int nYi, int nXj, int nYj)
- static double distance (double dXi, double dYi, double dXj, double dYj)
- static boolean boundingBoxesIntersect (double dXmin1, double dYmin1, double dXmax1, double dYmax1, double dXmin2, double dYmin2, double dXmax2, double dYmax2)
- static boolean isInside (int nX, int nY, int nT, int nR, int nB, int nL, int nTol)
- static boolean isInside (double dX, double dY, double dT, double dR, double dB, double dL, double dTol)

- static boolean isInBoundingBox (double dX, double dY, double dX1, double dY1, double dX2, double dY2)
- static double distAlongLine (double[ ] dPts, double[ ] dSeg, double dX, double dY)
- static boolean collinear (double dX1, double dY1, double dX2, double dY2, double dX3, double dY3)
- static double angle (double dX1, double dY1, double dX2, double dY2)
- static double angle (double dX1, double dY1, double dX2, double dY2, double dX3, double dY3)

Static Public Attributes

- static final double EARTH_MINOR_RADIUS = 6356752.0
- static final double EARTH_MAJOR_RADIUS = 6378137.0
- static final double EARTH_FLATTENING = EARTH_MINOR_RADIUS / EARTH_MAJOR_RADIUS

Private Member Functions

- Geo ()

7.7.1 Detailed Description

Definition at line 6 of file Geo.java.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 Geo()

```
cc.util.Geo.Geo ( )  [inline], [private]
```

Definition at line 13 of file Geo.java.

```
00014     {
00015     }
```

7.7.3 Member Function Documentation

7.7.3.1 angle() [1/2]

```
static double cc.util.Geo.angle (
            double dX1,
            double dY1,
            double dX2,
            double dY2 )  [inline], [static]
```

Definition at line 174 of file Geo.java.

```
00175     {
00176         return angle(dX1 + 1, dY1, dX1, dY1, dX2, dY2);
00177     }
```

References cc.util.Geo.angle().

Referenced by cc.util.Geo.angle().

Here is the call graph for this function:

Here is the caller graph for this function:

7.7.3.2 angle() [2/2]

```
static double cc.util.Geo.angle (
            double dX1,
            double dY1,
            double dX2,
            double dY2,
```

```
            double dX3,
            double dY3 )  [inline], [static]
```

Definition at line 180 of file Geo.java.

```
00181     {
00182         double dUi = dX1 - dX2;
00183         double dUj = dY1 - dY2;
00184         double dVi = dX3 - dX2;
00185         double dVj = dY3 - dY2;
00186         double dDot = dUi * dVi + dUj * dVj;
00187         double dLenU = Math.sqrt(dUi * dUi + dUj * dUj);
00188         double dLenV = Math.sqrt(dVi * dVi + dVj * dVj);
00189         if (dLenU == 0 || dLenV == 0) // prevent division by zero
00190             return Double.NaN;
00191         double dValue = dDot / (dLenU * dLenV);
00192         dValue = (dValue * 1000000000000L) / 1000000000000L; // round to help prevent value outside of
      the domain of arccos
00193         if (dValue > 1 || dValue < -1) // prevent domain error for arcos
00194             return Double.NaN;
00195
00196         return Math.acos(dValue); // return value in radians
00197     }
```

7.7.3.3 boundingBoxesIntersect()

```
static boolean cc.util.Geo.boundingBoxesIntersect (
            double dXmin1,
            double dYmin1,
            double dXmax1,
            double dYmax1,
            double dXmin2,
            double dYmin2,
            double dXmax2,
            double dYmax2 )  [inline], [static]
```

Definition at line 67 of file Geo.java.

```
00068     {
00069         return dYmax1 >= dYmin2 && dYmin1 <= dYmax2 && dXmax1 >= dXmin2 && dXmin1 <= dXmax2;
00070     }
```

7.7.3.4 collinear()

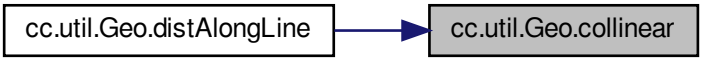
```
static boolean cc.util.Geo.collinear (
            double dX1,
            double dY1,
            double dX2,
            double dY2,
            double dX3,
            double dY3 )  [inline], [static]
```

Definition at line 168 of file Geo.java.

```
00169     {
00170         return Math.abs(dX1 * (dY2 - dY3) + dX2 * (dY3 - dY1) + dX3 * (dY1 - dY2)) < 0.0000001;
00171     }
```

Referenced by cc.util.Geo.distAlongLine().

Here is the caller graph for this function:



7.7.3.5 distAlongLine()

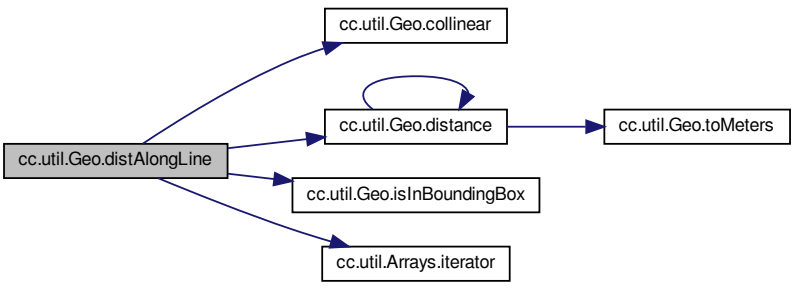
```
static double cc.util.Geo.distAlongLine (
            double[] dPts,
            double[] dSeg,
            double dX,
            double dY )  [inline], [static]
```

Definition at line 150 of file Geo.java.

```
00151     {
00152         double dDist = 0.0;
00153         Iterator<double[]> oIt = Arrays.iterator(dPts, dSeg, 1, 2);
00154         while (oIt.hasNext())
00155         {
00156             oIt.next();
00157             if (isInBoundingBox(dX, dY, dSeg[1], dSeg[0], dSeg[3], dSeg[2]) && collinear(dX, dY,
      dSeg[1], dSeg[0], dSeg[3], dSeg[2]))
00158             {
00159                 dDist += distance(dX, dY, dSeg[1], dSeg[0]);
00160                 return dDist;
00161             }
00162             dDist += distance(dSeg[1], dSeg[0], dSeg[3], dSeg[2]);
00163         }
00164         return Double.NaN;
00165     }
```

References cc.util.Geo.collinear(), cc.util.Geo.distance(), cc.util.Geo.isInBoundingBox(), and cc.util.Arrays.iterator().

Here is the call graph for this function:



7.7.3.6 distance() [1/2]

```
static double cc.util.Geo.distance (
            double dXi,
            double dYi,
            double dXj,
            double dYj )  [inline], [static]
```

Definition at line 54 of file Geo.java.

```
00055     {
00056         double dXd = dXj - dXi; // correct distance by latitude
00057 //      dXd = (dXd * Math.cos(Math.toRadians(dYi / 100000.0)) +
00058 //          dXd * Math.cos(Math.toRadians(dYj / 100000.0))) / 2.0;
00059 //      double dYd = (dYj - dYi) * EARTH_FLATTENING;
00060         double dYd = dYj - dYi;
00061         return Math.sqrt(dXd * dXd + dYd * dYd);
00062     }
```

7.7.3.7 distance() [2/2]

```
static double cc.util.Geo.distance (
            int nXi,
            int nYi,
            int nXj,
            int nYj )  [inline], [static]
```

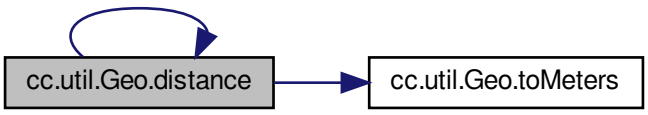
Definition at line 48 of file Geo.java.

```
00049     {
00050         return distance(toMeters(nXi), toMeters(nYi), toMeters(nXj), toMeters(nYj));
00051     }
```

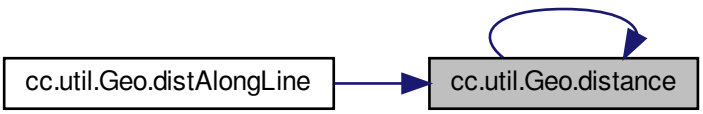
References cc.util.Geo.distance(), and cc.util.Geo.toMeters().

Referenced by cc.util.Geo.distAlongLine(), and cc.util.Geo.distance().

Here is the call graph for this function:



Here is the caller graph for this function:



7.7.3.8 fromIntDeg()

```
static double cc.util.Geo.fromIntDeg (
            int nOrd )  [inline], [static]
```

Definition at line 30 of file Geo.java.

```
00031     {
00032         return ((double)nOrd) / 10000000.0;
00033     }
```

7.7.3.9 getHash()

```
static int cc.util.Geo.getHash (
            int nLat,
            int nLon )  [inline], [static]
```

Definition at line 24 of file Geo.java.

```
00025     {
00026         return (scale(nLon) << 16) + scale(nLat);
00027     }
```

References cc.util.Geo.scale().

Here is the call graph for this function:

7.7.3.10 isInBoundingBox()

```
static boolean cc.util.Geo.isInBoundingBox (
            double dX,
            double dY,
            double dX1,
            double dY1,
            double dX2,
            double dY2 )  [inline], [static]
```

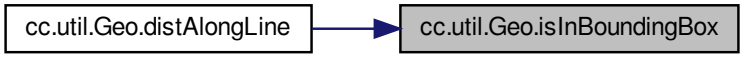
Definition at line 130 of file Geo.java.

```
00131     {
00132         if (dX1 > dX2)
00133         {
```

```
00134             double dTemp = dX1;
00135             dX1 = dX2;
00136             dX2 = dTemp;
00137         }
00138
00139         if (dY1 > dY2)
00140         {
00141             double dTemp = dY1;
00142             dY1 = dY2;
00143             dY2 = dTemp;
00144         }
00145
00146         return dX >= dX1 && dX <= dX2 && dY >= dY1 && dY <= dY2;
00147     }
```

Referenced by cc.util.Geo.distAlongLine().

Here is the caller graph for this function:



7.7.3.11 isInside() [1/2]

```
static boolean cc.util.Geo.isInside (
             double dX,
             double dY,
             double dT,
             double dR,
             double dB,
             double dL,
             double dTol )  [inline], [static]
```

Determines if the specified point is within the specified boundary. A specified tolerance adjusts the compared region as needed

Parameters

dX	x coordinate of point
dY	y coordinate of point
dT	y value of the top of the region
dR	x value of the right side of the region
dB	y value of the bottom of the region
dL	x value of the left side of the region
dTol	the allowed margin for a point to be considered inside

Returns

true if the point is inside or on the rectangular region

Definition at line 122 of file Geo.java.

```
00124     {
00125         return (dX >= dL - dTol && dX <= dR + dTol
00126            && dY >= dB - dTol && dY <= dT + dTol);
00127     }
```

7.7.3.12 isInside() [2/2]

```
static boolean cc.util.Geo.isInside (
             int nX,
             int nY,
             int nT,
             int nR,
             int nB,
             int nL,
             int nTol )  [inline], [static]
```

Determines if the specified point is within the specified boundary. A specified tolerance adjusts the compared region as needed.

Parameters

nX	x coordinate of point
nY	y coordinate of point
nT	y value of the top of the region
nR	x value of the right side of the region
nB	y value of the bottom of the region
nL	x value of the left side of the region
nTol	the allowed margin for a point to be considered inside

Returns

true if the point is inside or on the rectangular region

Definition at line 86 of file Geo.java.

```
00088     {
00089         if (nR < nL) // swap the left and right bounds as needed
00090         {
00091             nR ^= nL;
00092             nL ^= nR;
00093             nR ^= nL;
00094         }
00095
00096         if (nT < nB) // swap the top and bottom bounds as needed
00097         {
00098             nT ^= nB;
00099             nB ^= nT;
00100             nT ^= nB;
00101         }
00102
00103         // expand the bounds by the tolerance
00104         return (nX >= nL - nTol && nX <= nR + nTol
00105            && nY >= nB - nTol && nY <= nT + nTol);
00106     }
```

7.7.3.13 scale()

```
static int cc.util.Geo.scale (
            int nVal )  [inline], [static]
```

Definition at line 18 of file Geo.java.

```
00019     {
00020         return (int)Math.floor(((double)nVal) / 100000.0); // OSM geo-coordinates have 7 decimal
      places
00021     }
```

Referenced by cc.util.Geo.getHash().

Here is the caller graph for this function:

7.7.3.14 toIntDeg()

```
static int cc.util.Geo.toIntDeg (
            double dOrd )  [inline], [static]
```

Definition at line 36 of file Geo.java.

```
00037     {
00038         return (int)(dOrd * 10000000.0);
00039     }
```

7.7.3.15 toMeters()

```
static double cc.util.Geo.toMeters (
            int nOrd )  [inline], [static]
```

Definition at line 42 of file Geo.java.

```
00043     {
00044         return ((double)nOrd) / 100.0;
00045     }
```

Referenced by cc.util.Geo.distance().

Here is the caller graph for this function:

7.7.4 Member Data Documentation

7.7.4.1 EARTH_FLATTENING

```
final double cc.util.Geo.EARTH_FLATTENING = EARTH_MINOR_RADIUS / EARTH_MAJOR_RADIUS  [static]
```

Definition at line 10 of file Geo.java.

7.7.4.2 EARTH_MAJOR_RADIUS

```
final double cc.util.Geo.EARTH_MAJOR_RADIUS = 6378137.0  [static]
```

Definition at line 9 of file Geo.java.

7.7.4.3 EARTH_MINOR_RADIUS

```
final double cc.util.Geo.EARTH_MINOR_RADIUS = 6356752.0  [static]
```

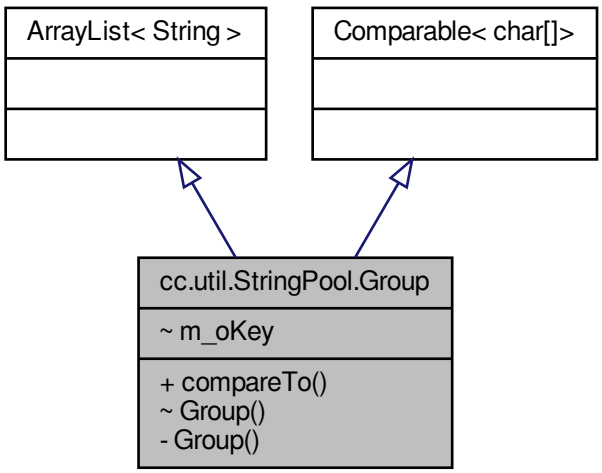
Definition at line 8 of file Geo.java.

The documentation for this class was generated from the following file:

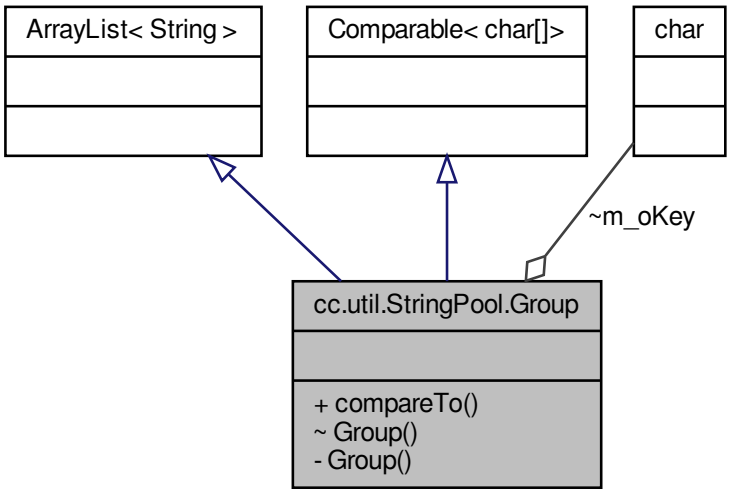
- src/cc/util/Geo.java

7.8 cc.util.StringPool.Group Class Reference

Inheritance diagram for cc.util.StringPool.Group:



Collaboration diagram for cc.util.StringPool.Group:



Public Member Functions

- int compareTo (char[ ] oRhs)

Package Functions

- Group (char[ ] oKey)

Package Attributes

- char[ ] m_oKey

Private Member Functions

- Group ()

7.8.1 Detailed Description

Definition at line 72 of file StringPool.java.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 Group() [1/2]

```
cc.util.StringPool.Group.Group ( )  [inline], [private]
```

Definition at line 77 of file StringPool.java.

```
00078         {
00079         }
```

7.8.2.2 Group() [2/2]

```
cc.util.StringPool.Group.Group (
             char[] oKey )  [inline], [package]
```

Definition at line 82 of file StringPool.java.

```
00083         {
00084             m_oKey = new char[]{oKey[0], oKey[1]};
00085         }
```

References cc.util.StringPool.Group.m_oKey.

7.8.3 Member Function Documentation

7.8.3.1 compareTo()

```
int cc.util.StringPool.Group.compareTo (
             char[] oRhs )  [inline]
```

Definition at line 89 of file StringPool.java.

```
00090         {
00091             int nComp = m_oKey[0] - oRhs[0];
00092             if (nComp == 0)
00093                 nComp = m_oKey[1] - oRhs[1];
00094
00095             return nComp;
00096         }
```

References cc.util.StringPool.Group.m_oKey.

7.8.4 Member Data Documentation

7.8.4.1 m_oKey

```
char [] cc.util.StringPool.Group.m_oKey  [package]
```

Definition at line 74 of file StringPool.java.

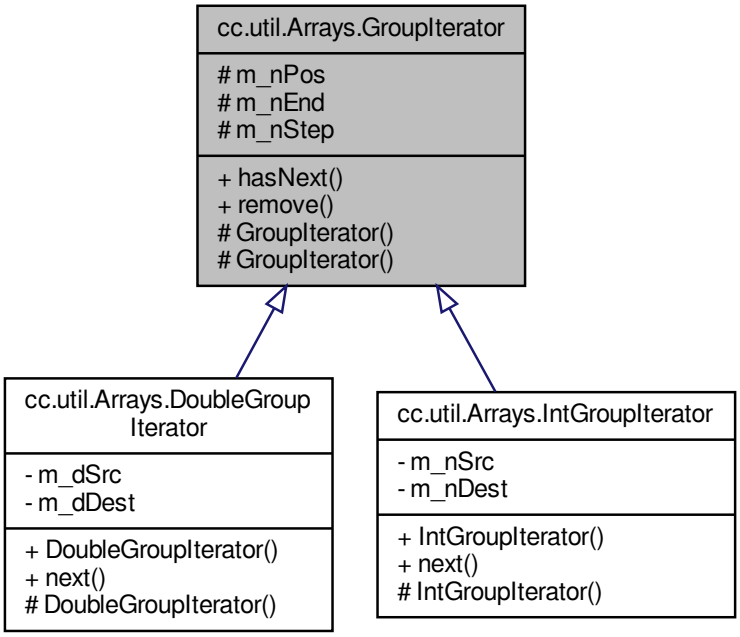
Referenced by cc.util.StringPool.Group.Group(), and cc.util.StringPool.Group.compareTo().

The documentation for this class was generated from the following file:

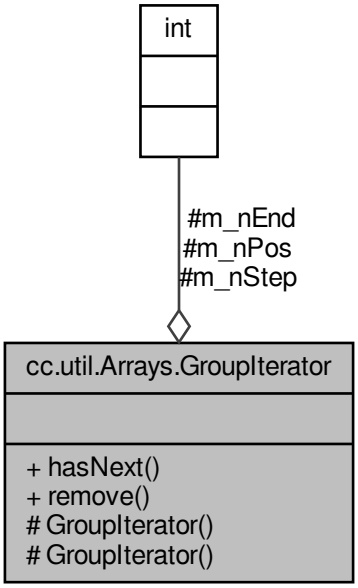
- src/cc/util/StringPool.java

7.9 cc.util.Arrays.GroupIterator Class Reference

Inheritance diagram for cc.util.Arrays.GroupIterator:



Collaboration diagram for cc.util.Arrays.GroupIterator:



Public Member Functions

- boolean hasNext ()
- void remove ()

Protected Member Functions

- GroupIterator ()
- GroupIterator (int nStart, int nLimit, int nDestSize, int nStep)

Protected Attributes

- int m_nPos
- int m_nEnd
- int m_nStep

7.9.1 Detailed Description

Definition at line 193 of file Arrays.java.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 GroupIterator() [1/2]

```
cc.util.Arrays.GroupIterator.GroupIterator ( )  [inline], [protected]
```

Definition at line 200 of file Arrays.java.

```
00201         {
00202         }
```

7.9.2.2 GroupIterator() [2/2]

```
cc.util.Arrays.GroupIterator.GroupIterator (
             int nStart,
             int nLimit,
             int nDestSize,
             int nStep )  [inline], [protected]
```

Definition at line 205 of file Arrays.java.

```
00206         {
00207             m_nStep = nStep;
00208             m_nEnd = nLimit - nDestSize; // array end boundary
00209             m_nPos = nStart;
00210         }
```

References cc.util.Arrays.GroupIterator.m_nEnd, cc.util.Arrays.GroupIterator.m_nPos, and cc.util.Arrays.GroupIterator.m_nStep.

7.9.3 Member Function Documentation

7.9.3.1 hasNext()

```
boolean cc.util.Arrays.GroupIterator.hasNext ( )  [inline]
```

Definition at line 213 of file Arrays.java.

```
00214         {
00215             return (m_nPos <= m_nEnd);
00216         }
```

References cc.util.Arrays.GroupIterator.m_nEnd, and cc.util.Arrays.GroupIterator.m_nPos.

7.9.3.2 remove()

```
void cc.util.Arrays.GroupIterator.remove ( )  [inline]
```

Definition at line 219 of file Arrays.java.

```
00220         {
00221             throw new UnsupportedOperationException("remove");
00222         }
```

7.9.4 Member Data Documentation

7.9.4.1 m_nEnd

```
int cc.util.Arrays.GroupIterator.m_nEnd  [protected]
```

Definition at line 196 of file Arrays.java.

Referenced by cc.util.Arrays.GroupIterator.GroupIterator(), and cc.util.Arrays.GroupIterator.hasNext().

7.9.4.2 m_nPos

```
int cc.util.Arrays.GroupIterator.m_nPos  [protected]
```

Definition at line 195 of file Arrays.java.

Referenced by cc.util.Arrays.GroupIterator.GroupIterator(), cc.util.Arrays.GroupIterator.hasNext(), cc.util.Arrays.DoubleGroupIterator.n
and cc.util.Arrays.IntGroupIterator.next().

7.9.4.3 m_nStep

```
int cc.util.Arrays.GroupIterator.m_nStep  [protected]
```

Definition at line 197 of file Arrays.java.

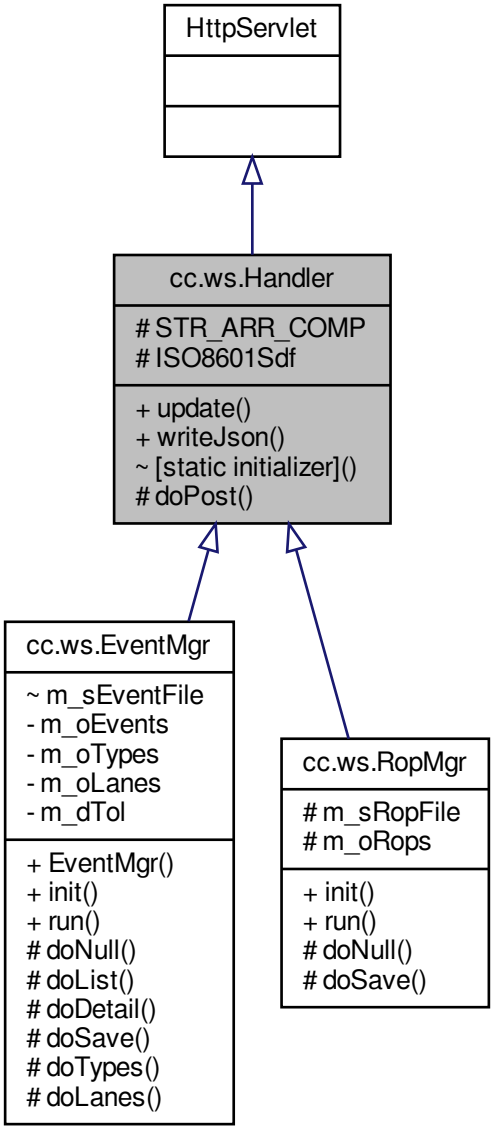
Referenced by cc.util.Arrays.GroupIterator.GroupIterator(), cc.util.Arrays.DoubleGroupIterator.next(), and cc.util.Arrays.IntGroupIterator.next().

The documentation for this class was generated from the following file:

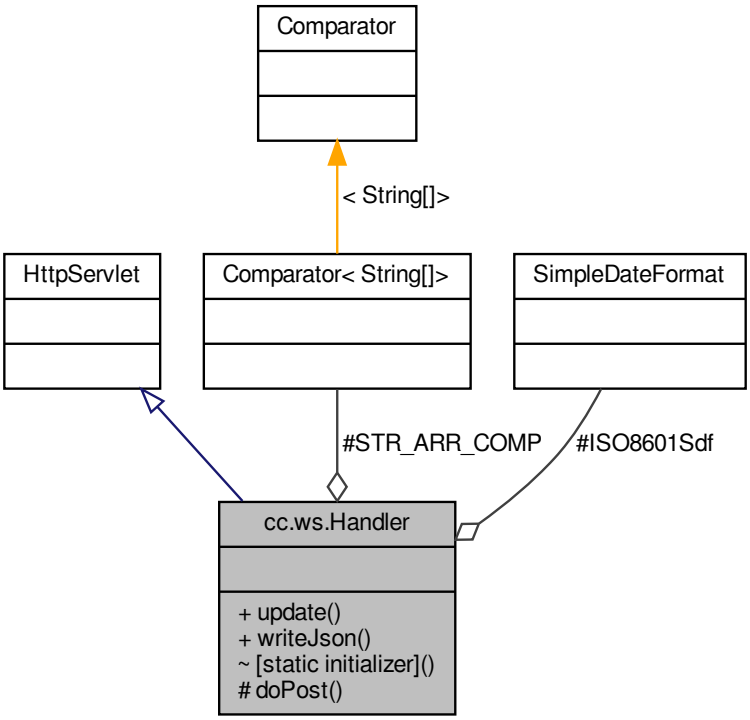
- src/cc/util/Arrays.java

7.10 cc.ws.Handler Class Reference

Inheritance diagram for cc.ws.Handler:



Collaboration diagram for cc.ws.Handler:



Static Public Member Functions

- static void update (String[ ] sObj, int nCols, CsvReader oIn)
- static void writeJson (PrintWriter oOut, String sId, String sData)

Protected Member Functions

- void doPost (HttpServletRequest oReq, HttpServletResponse oRep) throws ServletException, IOException

Static Protected Attributes

- static Comparator< String[ ]> STR_ARR_COMP = (String[ ] o1, String[ ] o2) -> o1[0].compareTo(o2[0])
- static final SimpleDateFormat ISO8601Sdf = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss'Z'")

Static Package Functions

- [static initializer]

7.10.1 Detailed Description

Definition at line 17 of file Handler.java.

7.10.2 Member Function Documentation

7.10.2.1 [static initializer]()

```
cc.ws.Handler.[static initializer]  [inline], [static], [package]
```

7.10.2.2 doPost()

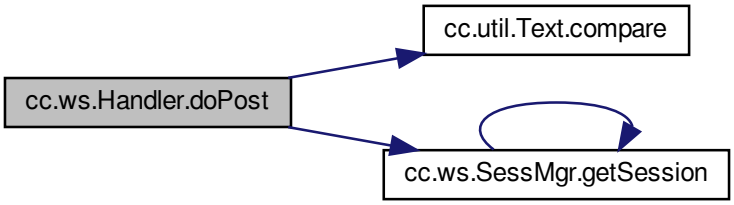
```
void cc.ws.Handler.doPost (
            HttpServletRequest oReq,
            HttpServletResponse oRep ) throws ServletException, IOException  [inline], [protected]
```

Definition at line 28 of file Handler.java.

```
00030     {
00031         try
00032         {
00033             Session oSess = SessMgr.getSession(oReq);
00034             if (oSess == null)
00035             {
00036                 oRep.sendError(401);
00037                 return;
00038             }
00039 
00040 
00041             StringBuilder sMethod = new StringBuilder("do");
00042             String sAction = oReq.getPathInfo();
00043             sMethod.append(sAction);
00044             if (sMethod.charAt(2) == '/') // remove leading "/"
00045                 sMethod.deleteCharAt(2);
00046 
00047             sMethod.setCharAt(2, Character.toUpperCase(sMethod.charAt(2))); // upper case the first
      character of the action
00048             if (Text.compare(sMethod, "doPost") == 0)
00049             {
00050                 oRep.sendError(401);
00051                 return;
00052             }
00053 
00054             for (Method oMethod : getClass().getDeclaredMethods())
00055             {
00056                 if (Text.compare(sMethod, oMethod.getName()) == 0)
00057                 {
00058                     try (PrintWriter oOut = oRep.getWriter())
00059                     {
00060                         oMethod.invoke(this, oSess, oReq, oOut);
00061                         return;
00062                     }
00063                 }
00064             }
00065             oRep.sendError(401);
00066         }
00067         catch (Exception oEx)
00068         {
00069             oRep.sendError(409);
00070         }
00071     }
```

References cc.util.Text.compare(), and cc.ws.SessMgr.getSession().

Here is the call graph for this function:



7.10.2.3 update()

```
static void cc.ws.Handler.update (
             String[] sObj,
             int nCols,
             CsvReader oIn )  [inline], [static]
```

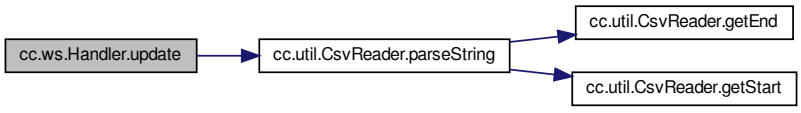
Definition at line 74 of file Handler.java.

```
00075     {
00076         int nLimit = Math.min(nCols, sObj.length);
00077         for (int i = 0; i < nLimit; i++)
00078             sObj[i] = oIn.parseString(i);
00079     }
```

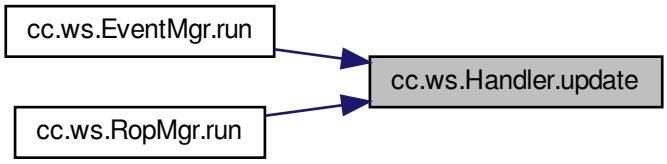
References cc.util.CsvReader.parseString().

Referenced by cc.ws.EventMgr.run(), and cc.ws.RopMgr.run().

Here is the call graph for this function:



Here is the caller graph for this function:



7.10.2.4 writeJson()

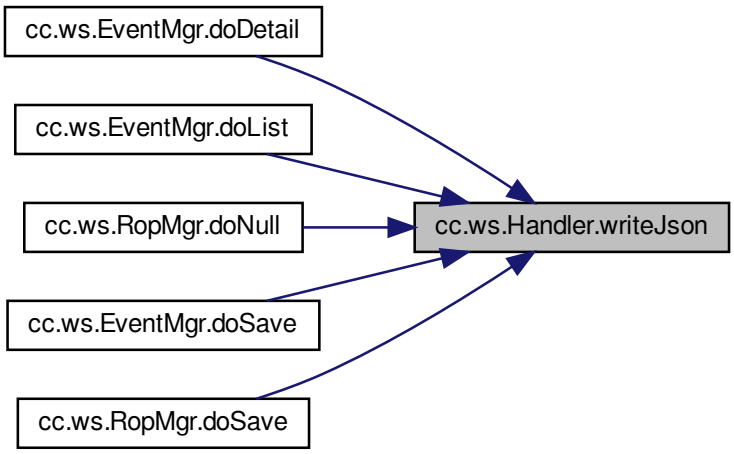
```
static void cc.ws.Handler.writeJson (
            PrintWriter oOut,
            String sId,
            String sData )  [inline], [static]
```

Definition at line 82 of file Handler.java.

```
00083     {
00084         oOut.write(String.format("\"%s\":%s", sId, sData));
00085     }
```

Referenced by cc.ws.EventMgr.doDetail(), cc.ws.EventMgr.doList(), cc.ws.RopMgr.doNull(), cc.ws.EventMgr.doSave(), and cc.ws.RopMgr.doSave().

Here is the caller graph for this function:



7.10.3 Member Data Documentation

7.10.3.1 ISO8601Sdf

```
final SimpleDateFormat cc.ws.Handler.ISO8601Sdf = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm↩
:ss'Z'")  [static], [protected]
```

Definition at line 20 of file Handler.java.

Referenced by cc.ws.EventMgr.doSave(), and cc.ws.RopMgr.doSave().

7.10.3.2 STR_ARR_COMP

```
Comparator<String[]> cc.ws.Handler.STR_ARR_COMP = (String[] o1, String[] o2) -> o1[0].compare↩
To(o2[0])  [static], [protected]
```

Definition at line 19 of file Handler.java.

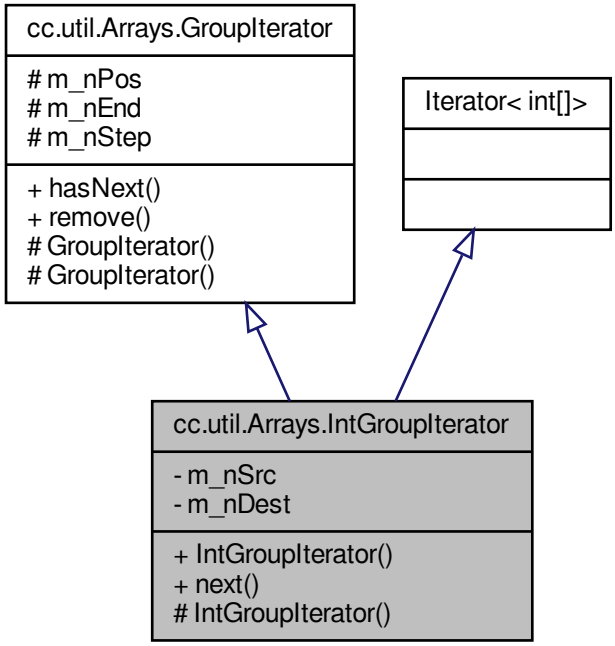
Referenced by cc.ws.EventMgr.doDetail(), cc.ws.EventMgr.doSave(), cc.ws.RopMgr.doSave(), cc.ws.EventMgr.run(), and cc.ws.RopMgr.run().

The documentation for this class was generated from the following file:

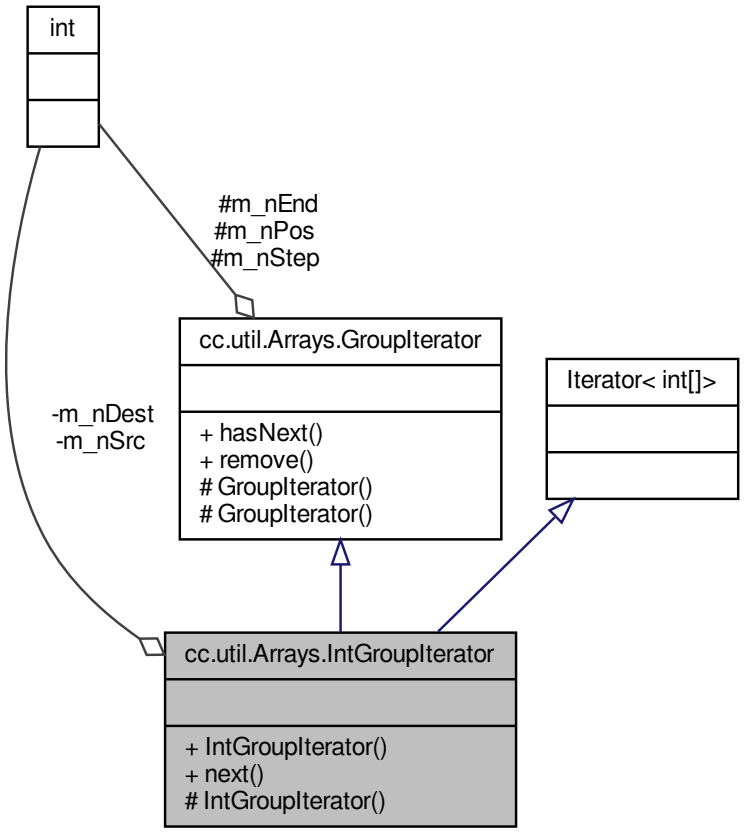
- src/cc/ws/Handler.java

7.11 cc.util.Arrays.IntGroupIterator Class Reference

Inheritance diagram for cc.util.Arrays.IntGroupIterator:



Collaboration diagram for cc.util.Arrays.IntGroupIterator:



Public Member Functions

- IntGroupIterator (int[ ] nSrc, int[ ] nDest, int nStart, int nStep) throws IllegalArgumentException
- int[ ] next ()

Protected Member Functions

- IntGroupIterator ()

Private Attributes

- int[ ] m_nSrc
- int[ ] m_nDest

Additional Inherited Members

7.11.1 Detailed Description

Definition at line 258 of file Arrays.java.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 IntGroupIterator() [1/2]

```
cc.util.Arrays.IntGroupIterator.IntGroupIterator ( )  [inline], [protected]
```

Definition at line 264 of file Arrays.java.

```
00265         {
00266         }
```

7.11.2.2 IntGroupIterator() [2/2]

```
cc.util.Arrays.IntGroupIterator.IntGroupIterator (
              int[] nSrc,
              int[] nDest,
              int nStart,
              int nStep ) throws IllegalArgumentException  [inline]
```

Definition at line 269 of file Arrays.java.

```
00271         {
00272             super(nStart, nSrc[0], nDest.length, nStep);
00273             if (nSrc.length == 0 || nDest.length == 0 || nSrc.length < nDest.length + 1 || nStart < 0
      || nStep <= 0)
00274                 throw new IllegalArgumentException();
00275             m_nDest = nDest;
00276             m_nSrc = nSrc; // local reference to values
00277         }
```

References cc.util.Arrays.IntGroupIterator.m_nDest, and cc.util.Arrays.IntGroupIterator.m_nSrc.

7.11.3 Member Function Documentation

7.11.3.1 next()

```
int[] cc.util.Arrays.IntGroupIterator.next ( )  [inline]
```

Definition at line 281 of file Arrays.java.

```
00282         {
00283             System.arraycopy(m_nSrc, m_nPos, m_nDest, 0, m_nDest.length);
00284             m_nPos += m_nStep; // shift to next group position
00285             return m_nDest;
00286         }
```

References cc.util.Arrays.IntGroupIterator.m_nDest, cc.util.Arrays.GroupIterator.m_nPos, cc.util.Arrays.IntGroupIterator.m_nSrc, and cc.util.Arrays.GroupIterator.m_nStep.

7.11.4 Member Data Documentation

7.11.4.1 m_nDest

```
int [] cc.util.Arrays.IntGroupIterator.m_nDest  [private]
```

Definition at line 261 of file Arrays.java.

Referenced by cc.util.Arrays.IntGroupIterator.IntGroupIterator(), and cc.util.Arrays.IntGroupIterator.next().

7.11.4.2 m_nSrc

```
int [] cc.util.Arrays.IntGroupIterator.m_nSrc  [private]
```

Definition at line 260 of file Arrays.java.

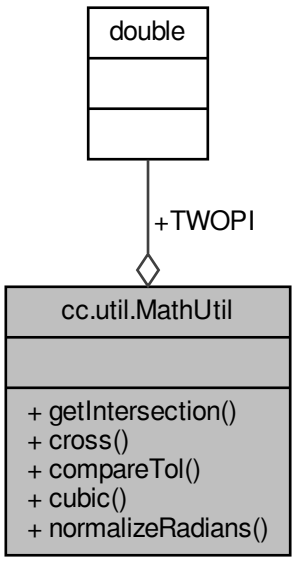
Referenced by cc.util.Arrays.IntGroupIterator.IntGroupIterator(), and cc.util.Arrays.IntGroupIterator.next().

The documentation for this class was generated from the following file:

- src/cc/util/Arrays.java

7.12 cc.util.MathUtil Class Reference

Collaboration diagram for cc.util.MathUtil:



Static Public Member Functions

- static void getIntersection (double dPx, double dPy, double dEnd1x, double dEnd1y, double dQx, double dQy, double dEnd2x, double dEnd2y, double[ ] dInter)
- static double cross (double dVx, double dVy, double dWx, double dWy)
- static int compareTol (double d1, double d2, double dTol)
- static double cubic (double dX, double dA, double dB, double dC, double dD)
- static double normalizeRadians (double dRad)

Static Public Attributes

- static double TWOPI = Math.PI ∗ 2

7.12.1 Detailed Description

Author

Federal Highway Administration

Definition at line 12 of file MathUtil.java.

7.12.2 Member Function Documentation

7.12.2.1 compareTol()

```
static int cc.util.MathUtil.compareTol (
            double d1,
            double d2,
            double dTol )  [inline], [static]
```

Definition at line 47 of file MathUtil.java.

```
00048     {
00049         if (d2 > d1)
00050         {
00051             if (d2 - d1 > dTol)
00052                 return -1;
00053         }
00054         else if (d1 - d2 > dTol)
00055             return 1;
00056         return 0;
00057     }
```

7.12.2.2 cross()

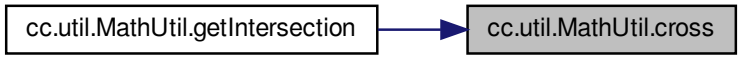
```
static double cc.util.MathUtil.cross (
            double dVx,
            double dVy,
            double dWx,
            double dWy )  [inline], [static]
```

Definition at line 41 of file MathUtil.java.

```
00042     {
00043         return dVx * dWy - dVy * dWx;
00044     }
```

Referenced by cc.util.MathUtil.getIntersection().

Here is the caller graph for this function:



7.12.2.3 cubic()

```
static double cc.util.MathUtil.cubic (
            double dX,
            double dA,
            double dB,
            double dC,
            double dD )  [inline], [static]
```

Definition at line 60 of file MathUtil.java.

```
00061     {
00062         return dA + (dB * dX) + (dC * dX * dX) + (dD * dX * dX * dX);
00063     }
```

7.12.2.4 getIntersection()

```
static void cc.util.MathUtil.getIntersection (
            double dPx,
            double dPy,
            double dEnd1x,
            double dEnd1y,
            double dQx,
            double dQy,
            double dEnd2x,
```

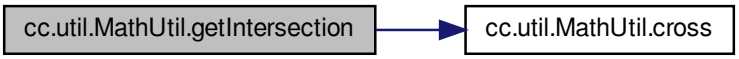
```
            double dEnd2y,
            double[] dInter )  [inline], [static]
```

Definition at line 17 of file MathUtil.java.

```
00018     {
00019         dInter[0] = Double.NaN;
00020         dInter[1] = Double.NaN;
00021         double dDeltaQPx = dQx - dPx;
00022         double dDeltaQPy = dQy - dPy;
00023         double dRx = dEnd1x - dPx;
00024         double dRy = dEnd1y - dPy;
00025         double dSx = dEnd2x - dQx;
00026         double dSy = dEnd2y - dQy;
00027         double dRCrossS = cross(dRx, dRy, dSx, dSy);
00028         if (dRCrossS == 0)
00029             return;
00030         double dT = cross(dDeltaQPx, dDeltaQPy, dSx, dSy) / dRCrossS;
00031         if (dT < 0 || dT > 1)
00032             return;
00033         double dU = cross(dDeltaQPx, dDeltaQPy, dRx, dRy) / dRCrossS;
00034         if (dU < 0 || dU > 1)
00035             return;
00036         dInter[0] = dPx + dT * dRx;
00037         dInter[1] = dPy + dT * dRy;
00038     }
```

References cc.util.MathUtil.cross().

Here is the call graph for this function:



7.12.2.5 normalizeRadians()

```
static double cc.util.MathUtil.normalizeRadians (
            double dRad )  [inline], [static]
```

Definition at line 66 of file MathUtil.java.

```
00067     {
00068         if (dRad < 0)
00069             return dRad + Math.ceil(-dRad / TWOPI) * TWOPI;
00070         else if (dRad >= TWOPI)
00071             return dRad - Math.floor(dRad / TWOPI) * TWOPI;
00072 
00073         return dRad;
00074     }
```

References cc.util.MathUtil.TWOPI.

7.12.3 Member Data Documentation

7.12.3.1 TWOPI

```
double cc.util.MathUtil.TWOPI = Math.PI * 2  [static]
```

Definition at line 14 of file MathUtil.java.

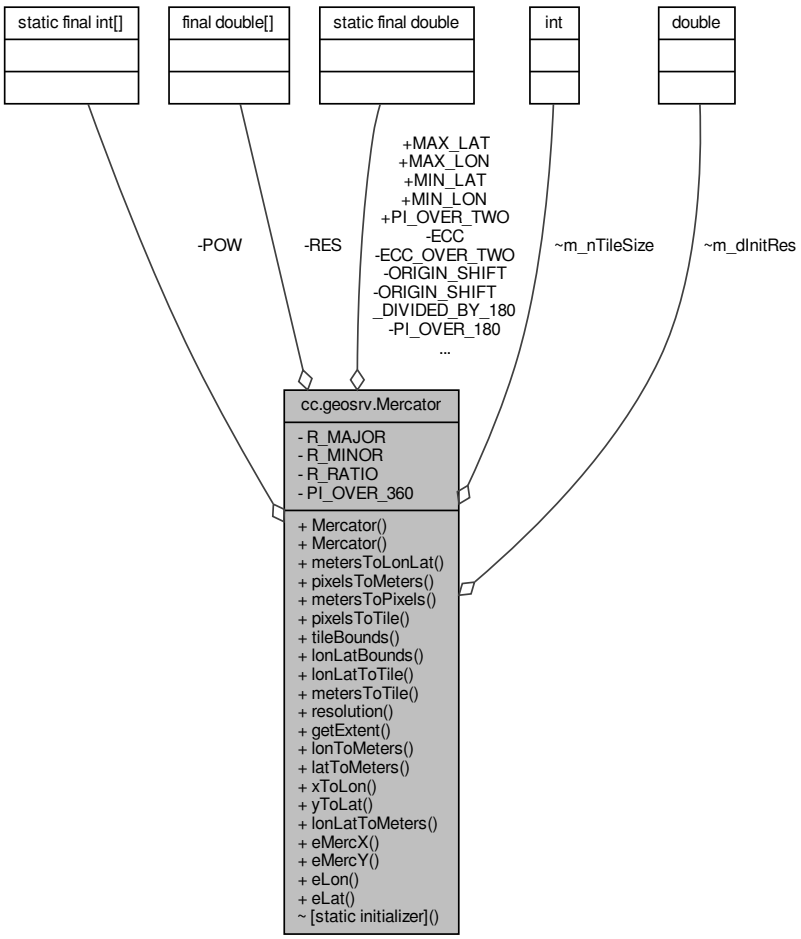
Referenced by cc.util.MathUtil.normalizeRadians().

The documentation for this class was generated from the following file:

- src/cc/util/MathUtil.java

7.13 cc.geosrv.Mercator Class Reference

Collaboration diagram for cc.geosrv.Mercator:



Public Member Functions

- Mercator ()
- Mercator (int nTileSize)
- void metersToLonLat (double dX, double dY, double[ ] dLatLon)
- void pixelsToMeters (double dXp, double dYp, int nZoom, double[ ] dMeters)
- void metersToPixels (double dXm, double dYm, int nZoom, double[ ] dPixels)
- void pixelsToTile (double dXp, double dYp, int[ ] nTiles)
- void tileBounds (double dXt, double dYt, int nZoom, double[ ] dBounds)
- void lonLatBounds (double dXt, double dYt, int nZoom, double[ ] dBounds)
- void lonLatToTile (double dLon, double dLat, int nZoom, int[ ] nTiles)
- void metersToTile (double dXm, double dYm, int nZoom, int[ ] nTiles)
- double resolution (int nZoom)

Static Public Member Functions

- static int getExtent (int nZoom)
- static double lonToMeters (double dLon)
- static double latToMeters (double dLat)
- static double xToLon (double dX)
- static double yToLat (double dY)
- static void lonLatToMeters (double dLon, double dLat, double[ ] dMeters)
- static double eMercX (double lon)
- static double eMercY (double lat)
- static double eLon (double dX)
- static double eLat (double dY)

Static Public Attributes

- static final double PI_OVER_TWO = Math.PI / 2.0
- static final double MAX_LAT = 85.05112877980659
- static final double MIN_LAT = -MAX_LAT
- static final double MAX_LON = 180
- static final double MIN_LON = -MAX_LON

Static Package Functions

- [static initializer]

Package Attributes

- int m_nTileSize
- double m_dInitRes

Private Attributes

- final double[ ] RES = new double[24]

Static Private Attributes

- static final int[ ] POW = new int[24]
- static final double R_MAJOR = 6378137.0
- static final double R_MINOR = 6356752.3142
- static final double R_RATIO = R_MINOR / R_MAJOR
- static final double ECC = Math.sqrt(1.0 - (R_RATIO ∗ R_RATIO))
- static final double ECC_OVER_TWO = ECC / 2.0
- static final double ORIGIN_SHIFT = Math.PI ∗ R_MAJOR
- static final double ORIGIN_SHIFT_DIVIDED_BY_180 = ORIGIN_SHIFT / 180.0
- static final double PI_OVER_180 = Math.PI / 180.0
- static final double PI_OVER_360 = PI_OVER_180 / 2.0

7.13.1 Detailed Description

Author

Federal Highway Administration

Definition at line 13 of file Mercator.java.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 Mercator() [1/2]

```
cc.geosrv.Mercator.Mercator ( )  [inline]
```

Definition at line 43 of file Mercator.java.

```
00044     {
00045         this(256);
00046     }
```

7.13.2.2 Mercator() [2/2]

```
cc.geosrv.Mercator.Mercator (
            int nTileSize )  [inline]
```

Definition at line 49 of file Mercator.java.

```
00050     {
00051         m_nTileSize = nTileSize;
00052         m_dInitRes = 2.0 * ORIGIN_SHIFT / m_nTileSize;
00053         for (int nIndex = 0; nIndex < RES.length; nIndex++)
00054             RES[nIndex] = m_dInitRes / POW[nIndex];
00055     }
```

References cc.geosrv.Mercator.m_dInitRes, cc.geosrv.Mercator.m_nTileSize, cc.geosrv.Mercator.ORIGIN_SHIFT, cc.geosrv.Mercator.POW, and cc.geosrv.Mercator.RES.

7.13.3 Member Function Documentation

7.13.3.1 [static initializer]()

```
cc.geosrv.Mercator.[static initializer]  [inline], [static], [package]
```

References cc.geosrv.Mercator.POW.

7.13.3.2 eLat()

```
static double cc.geosrv.Mercator.eLat (
            double dY )  [inline], [static]
```

Definition at line 200 of file Mercator.java.

```
00201     {
00202         double ts = Math.exp(-dY / R_MAJOR);
00203         double phi = PI_OVER_TWO - 2 * Math.atan(ts);
00204         double dphi = 1.0;
00205         for (int i = 0; Math.abs(dphi) > 0.000000001 && i < 15; i++)
00206         {
00207             double con = ECC * Math.sin(phi);
00208             dphi = PI_OVER_TWO - 2 * Math.atan(ts * Math.pow((1.0 - con) / (1.0 + con), ECC_OVER_TWO))
      - phi;
00209             phi += dphi;
00210         }
00211
00212         return Math.toDegrees(phi);
00213     }
```

References cc.geosrv.Mercator.ECC, cc.geosrv.Mercator.ECC_OVER_TWO, cc.geosrv.Mercator.PI_OVER_TWO, and cc.geosrv.Mercator.R_MAJOR.

7.13.3.3 eLon()

```
static double cc.geosrv.Mercator.eLon (
            double dX )  [inline], [static]
```

Definition at line 194 of file Mercator.java.

```
00195     {
00196         return Math.toDegrees(dX / R_MAJOR);
00197     }
```

References cc.geosrv.Mercator.R_MAJOR.

7.13.3.4 eMercX()

```
static double cc.geosrv.Mercator.eMercX (
            double lon )  [inline], [static]
```

Definition at line 174 of file Mercator.java.

```
00174                                           {
00175         return R_MAJOR * Math.toRadians(lon);
00176     }
```

References cc.geosrv.Mercator.R_MAJOR.

7.13.3.5 eMercY()

```
static double cc.geosrv.Mercator.eMercY (
            double lat )  [inline], [static]
```

Definition at line 178 of file Mercator.java.

```
00178                                           {
00179         if (lat > 89.5) {
00180             lat = 89.5;
00181         }
00182         if (lat < -89.5) {
00183             lat = -89.5;
00184         }
00185         double phi = Math.toRadians(lat);
00186         double con = ECC * Math.sin(phi);
00187         con = Math.pow(((1.0-con)/(1.0+con)), ECC_OVER_TWO);
00188         double ts = Math.tan(0.5 * ((PI_OVER_TWO) - phi))/con;
00189         double y = 0 - R_MAJOR * Math.log(ts);
00190         return y;
00191     }
```

References cc.geosrv.Mercator.ECC, cc.geosrv.Mercator.ECC_OVER_TWO, cc.geosrv.Mercator.PI_OVER_TWO, and cc.geosrv.Mercator.R_MAJOR.

7.13.3.6 getExtent()

```
static int cc.geosrv.Mercator.getExtent (
            int nZoom )  [inline], [static]
```

Definition at line 58 of file Mercator.java.

```
00059     {
00060         return POW[nZoom] * 256;
00061     }
```

References cc.geosrv.Mercator.POW.

7.13.3.7 latToMeters()

```
static double cc.geosrv.Mercator.latToMeters (
            double dLat )  [inline], [static]
```

Definition at line 70 of file Mercator.java.

```
00071     {
00072         return Math.log(Math.tan((90.0 + dLat) * PI_OVER_360)) * R_MAJOR;
00073     }
```

References cc.geosrv.Mercator.PI_OVER_360, and cc.geosrv.Mercator.R_MAJOR.

Referenced by cc.geosrv.Mercator.lonLatToMeters().

Here is the caller graph for this function:

7.13.3.8 lonLatBounds()

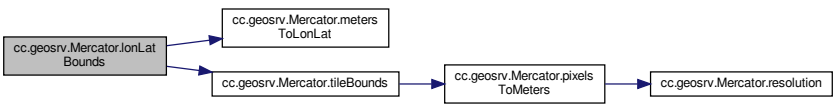
```
void cc.geosrv.Mercator.lonLatBounds (
            double dXt,
            double dYt,
            int nZoom,
            double[] dBounds )  [inline]
```

Definition at line 134 of file Mercator.java.

```
00135     {
00136         double[] dMeterBounds = new double[4];
00137         tileBounds(dXt, dYt, nZoom, dMeterBounds);
00138         double[] dLonLat = new double[2];
00139         metersToLonLat(dMeterBounds[0], dMeterBounds[1], dLonLat);
00140         dBounds[0] = dLonLat[0];
00141         dBounds[1] = dLonLat[1];
00142         metersToLonLat(dMeterBounds[2], dMeterBounds[3], dLonLat);
00143         dBounds[2] = dLonLat[0];
00144         dBounds[3] = dLonLat[1];
00145     }
```

References cc.geosrv.Mercator.metersToLonLat(), and cc.geosrv.Mercator.tileBounds().

Here is the call graph for this function:



7.13.3.9 lonLatToMeters()

```
static void cc.geosrv.Mercator.lonLatToMeters (
            double dLon,
            double dLat,
            double[] dMeters )  [inline], [static]
```

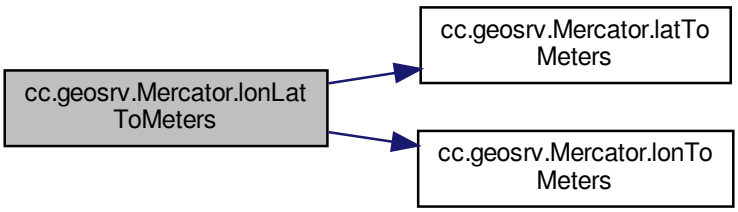
Definition at line 89 of file Mercator.java.

```
00090     {
00091         dMeters[0] = lonToMeters(dLon);
00092         dMeters[1] = latToMeters(dLat);
00093     }
```

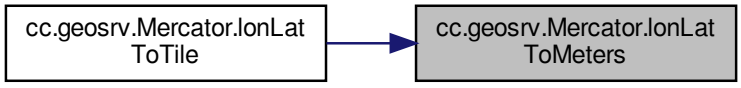
References cc.geosrv.Mercator.latToMeters(), and cc.geosrv.Mercator.lonToMeters().

Referenced by cc.geosrv.Mercator.lonLatToTile().

Here is the call graph for this function:



Here is the caller graph for this function:



7.13.3.10 lonLatToTile()

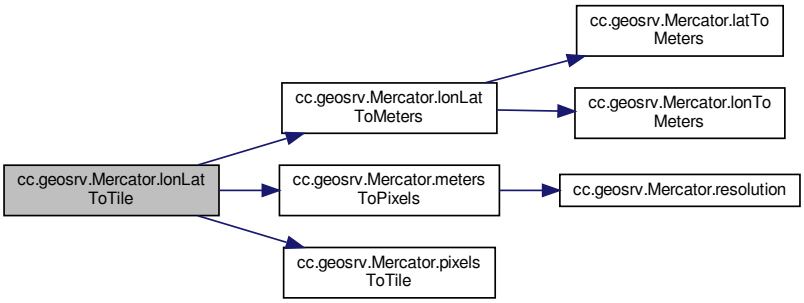
```
void cc.geosrv.Mercator.lonLatToTile (
            double dLon,
            double dLat,
            int nZoom,
            int[] nTiles )  [inline]
```

Definition at line 148 of file Mercator.java.

```
00149     {
00150         double[] dTemp = new double[2];
00151         lonLatToMeters(dLon, dLat, dTemp);
00152         metersToPixels(dTemp[0], dTemp[1], nZoom, dTemp);
00153         pixelsToTile(dTemp[0], dTemp[1], nTiles);
00154         nTiles[1] = POW[nZoom] - nTiles[1] - 1;
00155     }
```

References cc.geosrv.Mercator.lonLatToMeters(), cc.geosrv.Mercator.metersToPixels(), cc.geosrv.Mercator.pixelsToTile(), and cc.geosrv.Mercator.POW.

Here is the call graph for this function:



7.13.3.11 lonToMeters()

```
static double cc.geosrv.Mercator.lonToMeters (
            double dLon )  [inline], [static]
```

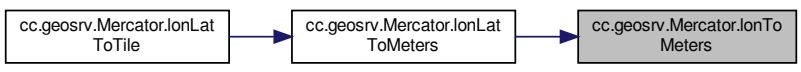
Definition at line 64 of file Mercator.java.

```
00065     {
00066         return dLon * ORIGIN_SHIFT_DIVIDED_BY_180;
00067     }
```

References cc.geosrv.Mercator.ORIGIN_SHIFT_DIVIDED_BY_180.

Referenced by cc.geosrv.Mercator.lonLatToMeters().

Here is the caller graph for this function:



7.13.3.12 metersToLonLat()

```
void cc.geosrv.Mercator.metersToLonLat (
            double dX,
            double dY,
            double[] dLatLon )  [inline]
```

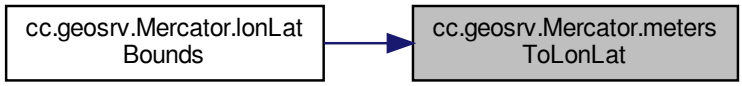
Definition at line 96 of file Mercator.java.

```
00097     {
00098         dLatLon[0] = dX / ORIGIN_SHIFT * 180.0;
00099         dLatLon[1] = dY / ORIGIN_SHIFT * 180.0;
00100         dLatLon[1] = 180.0 / Math.PI * (2 * Math.atan(Math.exp(dLatLon[1] * PI_OVER_180)) -
      PI_OVER_TWO);
00101     }
```

References cc.geosrv.Mercator.ORIGIN_SHIFT, cc.geosrv.Mercator.PI_OVER_180, and cc.geosrv.Mercator.PI_OVER_TWO.

Referenced by cc.geosrv.Mercator.lonLatBounds().

Here is the caller graph for this function:



7.13.3.13 metersToPixels()

```
void cc.geosrv.Mercator.metersToPixels (
            double dXm,
            double dYm,
            int nZoom,
            double[] dPixels )  [inline]
```

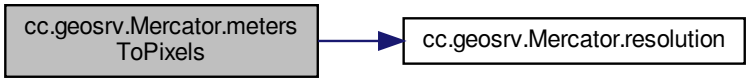
Definition at line 110 of file Mercator.java.

```
00111     {
00112         double dRes = resolution(nZoom);
00113         dPixels[0] = (dXm + ORIGIN_SHIFT) / dRes;
00114         dPixels[1] = (dYm + ORIGIN_SHIFT) / dRes;
00115     }
```

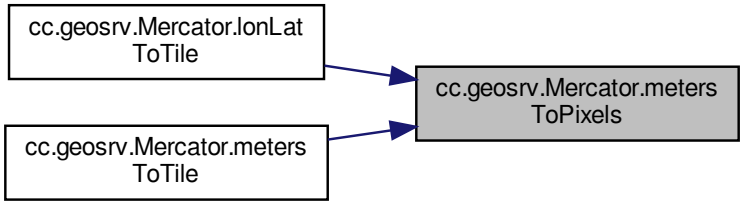
References cc.geosrv.Mercator.ORIGIN_SHIFT, and cc.geosrv.Mercator.resolution().

Referenced by cc.geosrv.Mercator.lonLatToTile(), and cc.geosrv.Mercator.metersToTile().

Here is the call graph for this function:



Here is the caller graph for this function:



7.13.3.14 metersToTile()

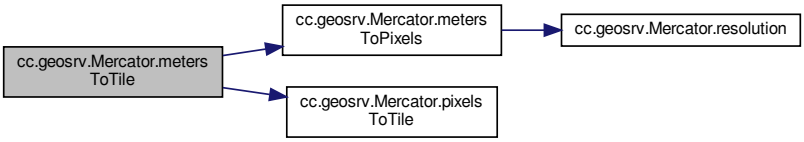
```
void cc.geosrv.Mercator.metersToTile (
            double dXm,
            double dYm,
            int nZoom,
            int[] nTiles )  [inline]
```

Definition at line 158 of file Mercator.java.

```
00159     {
00160         double[] dPixels = new double[2];
00161         metersToPixels(dXm, dYm, nZoom, dPixels);
00162         pixelsToTile(dPixels[0], dPixels[1], nTiles);
00163         nTiles[1] = POW[nZoom] - nTiles[1] - 1;
00164     }
```

References cc.geosrv.Mercator.metersToPixels(), cc.geosrv.Mercator.pixelsToTile(), and cc.geosrv.Mercator.POW.

Here is the call graph for this function:



7.13.3.15 pixelsToMeters()

```
void cc.geosrv.Mercator.pixelsToMeters (
            double dXp,
            double dYp,
            int nZoom,
            double[] dMeters )  [inline]
```

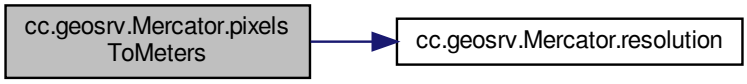
Definition at line 103 of file Mercator.java.

```
00104     {
00105         double dRes = resolution(nZoom);
00106         dMeters[0] = dXp * dRes - ORIGIN_SHIFT;
00107         dMeters[1] = -(dYp * dRes - ORIGIN_SHIFT);
00108     }
```

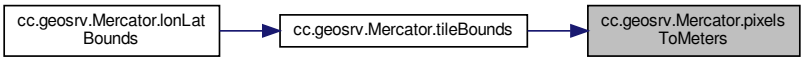
References cc.geosrv.Mercator.ORIGIN_SHIFT, and cc.geosrv.Mercator.resolution().

Referenced by cc.geosrv.Mercator.tileBounds().

Here is the call graph for this function:



Here is the caller graph for this function:



7.13.3.16 pixelsToTile()

```
void cc.geosrv.Mercator.pixelsToTile (
            double dXp,
            double dYp,
            int[] nTiles )  [inline]
```

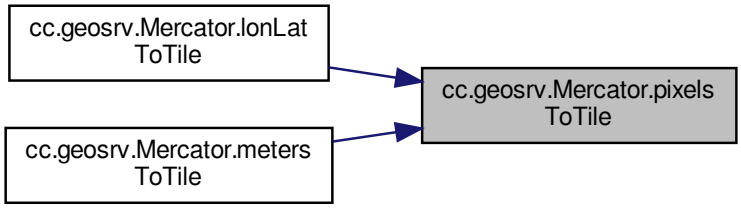
Definition at line 117 of file Mercator.java.

```
00118     {
00119         nTiles[0] = (int)((Math.ceil(dXp / m_nTileSize)) - 1);
00120         nTiles[1] = (int)((Math.ceil(dYp / m_nTileSize)) - 1);
00121     }
```

References cc.geosrv.Mercator.m_nTileSize.

Referenced by cc.geosrv.Mercator.lonLatToTile(), and cc.geosrv.Mercator.metersToTile().

Here is the caller graph for this function:



7.13.3.17 resolution()

```
double cc.geosrv.Mercator.resolution (
            int nZoom )  [inline]
```

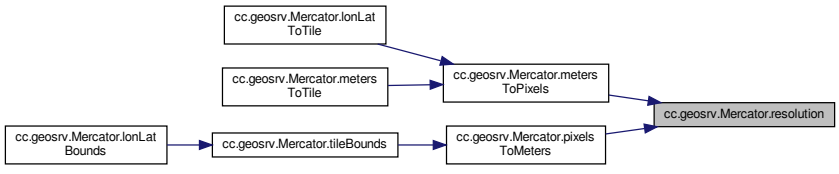
Definition at line 167 of file Mercator.java.

```
00168     {
00169 //      return m_dInitRes / POW[nZoom];
00170         return RES[nZoom];
00171     }
```

References cc.geosrv.Mercator.RES.

Referenced by cc.geosrv.Mercator.metersToPixels(), and cc.geosrv.Mercator.pixelsToMeters().

Here is the caller graph for this function:



7.13.3.18 tileBounds()

```
void cc.geosrv.Mercator.tileBounds (
            double dXt,
            double dYt,
            int nZoom,
            double[] dBounds )  [inline]
```

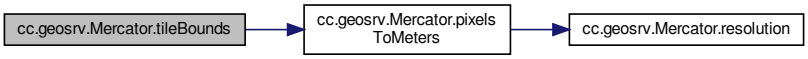
Definition at line 123 of file Mercator.java.

```
00124     {
00125         double[] dMeters = new double[2];
00126         pixelsToMeters(dXt * m_nTileSize, dYt * m_nTileSize, nZoom, dMeters);
00127         dBounds[0] = dMeters[0];
00128         dBounds[3] = dMeters[1];
00129         pixelsToMeters((dXt + 1) * m_nTileSize, (dYt + 1) * m_nTileSize, nZoom, dMeters);
00130         dBounds[2] = dMeters[0];
00131         dBounds[1] = dMeters[1];
00132     }
```

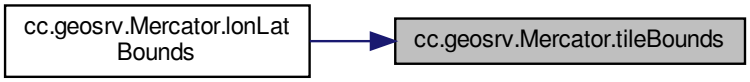
References cc.geosrv.Mercator.m_nTileSize, and cc.geosrv.Mercator.pixelsToMeters().

Referenced by cc.geosrv.Mercator.lonLatBounds().

Here is the call graph for this function:



Here is the caller graph for this function:



7.13.3.19 xToLon()

```
static double cc.geosrv.Mercator.xToLon (
            double dX )  [inline], [static]
```

Definition at line 76 of file Mercator.java.

```
00077     {
00078         return dX / ORIGIN_SHIFT * 180.0;
00079     }
```

References cc.geosrv.Mercator.ORIGIN_SHIFT.

7.13.3.20 yToLat()

```
static double cc.geosrv.Mercator.yToLat (
            double dY )  [inline], [static]
```

Definition at line 82 of file Mercator.java.

```
00083     {
00084         double dLat = dY / ORIGIN_SHIFT * 180.0;
00085         return 180.0 / Math.PI * (2 * Math.atan(Math.exp(dLat * PI_OVER_180)) - PI_OVER_TWO);
00086     }
```

References cc.geosrv.Mercator.ORIGIN_SHIFT, cc.geosrv.Mercator.PI_OVER_180, and cc.geosrv.Mercator.PI_OVER_TWO.

7.13.4 Member Data Documentation

7.13.4.1 ECC

```
final double cc.geosrv.Mercator.ECC = Math.sqrt(1.0 - (R_RATIO * R_RATIO))  [static], [private]
```

Definition at line 20 of file Mercator.java.

Referenced by cc.geosrv.Mercator.eLat(), and cc.geosrv.Mercator.eMercY().

7.13.4.2 ECC_OVER_TWO

```
final double cc.geosrv.Mercator.ECC_OVER_TWO = ECC / 2.0  [static], [private]
```

Definition at line 21 of file Mercator.java.

Referenced by cc.geosrv.Mercator.eLat(), and cc.geosrv.Mercator.eMercY().

7.13.4.3 m_dInitRes

```
double cc.geosrv.Mercator.m_dInitRes  [package]
```

Definition at line 33 of file Mercator.java.

Referenced by cc.geosrv.Mercator.Mercator().

7.13.4.4 m_nTileSize

```
int cc.geosrv.Mercator.m_nTileSize  [package]
```

Definition at line 32 of file Mercator.java.

Referenced by cc.geosrv.Mercator.Mercator(), cc.geosrv.Mercator.pixelsToTile(), and cc.geosrv.Mercator.tileBounds().

7.13.4.5 MAX_LAT

```
final double cc.geosrv.Mercator.MAX_LAT = 85.05112877980659  [static]
```

Definition at line 27 of file Mercator.java.

7.13.4.6 MAX_LON

```
final double cc.geosrv.Mercator.MAX_LON = 180  [static]
```

Definition at line 29 of file Mercator.java.

7.13.4.7 MIN_LAT

```
final double cc.geosrv.Mercator.MIN_LAT = -MAX_LAT  [static]
```

Definition at line 28 of file Mercator.java.

7.13.4.8 MIN_LON

```
final double cc.geosrv.Mercator.MIN_LON = -MAX_LON  [static]
```

Definition at line 30 of file Mercator.java.

7.13.4.9 ORIGIN_SHIFT

```
final double cc.geosrv.Mercator.ORIGIN_SHIFT = Math.PI * R_MAJOR  [static], [private]
```

Definition at line 23 of file Mercator.java.

Referenced by cc.geosrv.Mercator.Mercator(), cc.geosrv.Mercator.metersToLonLat(), cc.geosrv.Mercator.metersToPixels(), cc.geosrv.Mercator.pixelsToMeters(), cc.geosrv.Mercator.xToLon(), and cc.geosrv.Mercator.yToLat().

7.13.4.10 ORIGIN_SHIFT_DIVIDED_BY_180

```
final double cc.geosrv.Mercator.ORIGIN_SHIFT_DIVIDED_BY_180 = ORIGIN_SHIFT / 180.0  [static],
[private]
```

Definition at line 24 of file Mercator.java.

Referenced by cc.geosrv.Mercator.lonToMeters().

7.13.4.11 PI_OVER_180

```
final double cc.geosrv.Mercator.PI_OVER_180 = Math.PI / 180.0  [static], [private]
```

Definition at line 25 of file Mercator.java.

Referenced by cc.geosrv.Mercator.metersToLonLat(), and cc.geosrv.Mercator.yToLat().

7.13.4.12 PI_OVER_360

```
final double cc.geosrv.Mercator.PI_OVER_360 = PI_OVER_180 / 2.0  [static], [private]
```

Definition at line 26 of file Mercator.java.

Referenced by cc.geosrv.Mercator.latToMeters().

7.13.4.13 PI_OVER_TWO

```
final double cc.geosrv.Mercator.PI_OVER_TWO = Math.PI / 2.0  [static]
```

Definition at line 22 of file Mercator.java.

Referenced by cc.geosrv.Mercator.eLat(), cc.geosrv.Mercator.eMercY(), cc.geosrv.Mercator.metersToLonLat(), and cc.geosrv.Mercator.yToLat().

7.13.4.14 POW

```
final int [] cc.geosrv.Mercator.POW = new int[24]  [static], [private]
```

Definition at line 15 of file Mercator.java.

Referenced by cc.geosrv.Mercator.Mercator(), cc.geosrv.Mercator.[static initializer](), cc.geosrv.Mercator.getExtent(), cc.geosrv.Mercator.lonLatToTile(), and cc.geosrv.Mercator.metersToTile().

7.13.4.15 R_MAJOR

```
final double cc.geosrv.Mercator.R_MAJOR = 6378137.0  [static], [private]
```

Definition at line 17 of file Mercator.java.

Referenced by cc.geosrv.Mercator.eLat(), cc.geosrv.Mercator.eLon(), cc.geosrv.Mercator.eMercX(), cc.geosrv.Mercator.eMercY(), and cc.geosrv.Mercator.latToMeters().

7.13.4.16 R_MINOR

```
final double cc.geosrv.Mercator.R_MINOR = 6356752.3142  [static], [private]
```

Definition at line 18 of file Mercator.java.

7.13.4.17 R_RATIO

```
final double cc.geosrv.Mercator.R_RATIO = R_MINOR / R_MAJOR  [static], [private]
```

Definition at line 19 of file Mercator.java.

7.13.4.18 RES

```
final double [] cc.geosrv.Mercator.RES = new double[24]  [private]
```

Definition at line 16 of file Mercator.java.

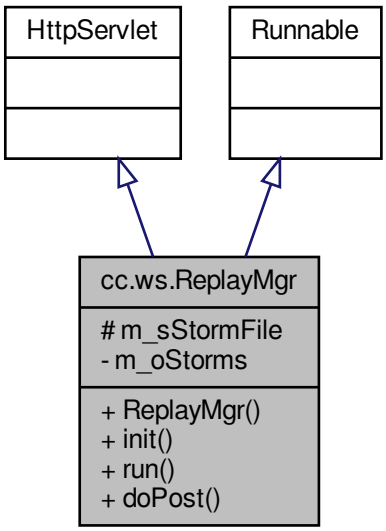
Referenced by cc.geosrv.Mercator.Mercator(), and cc.geosrv.Mercator.resolution().

The documentation for this class was generated from the following file:

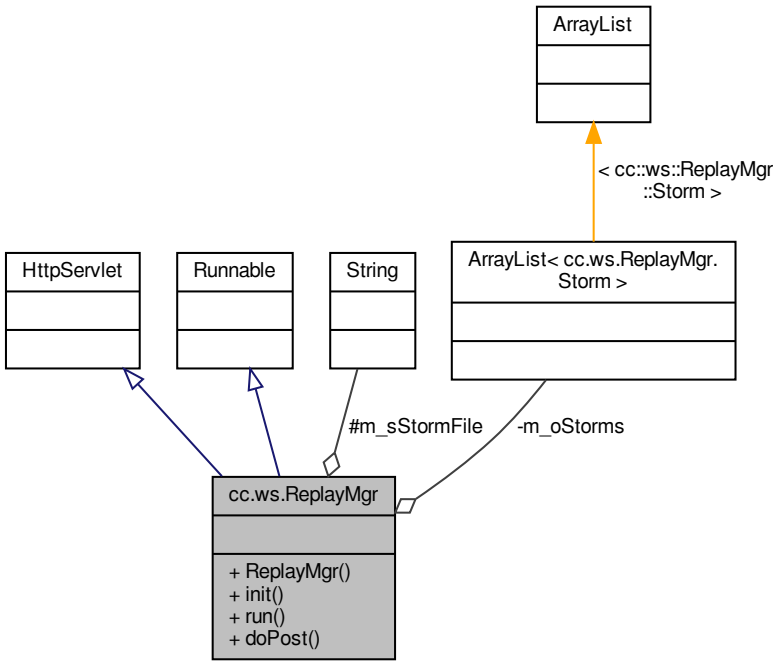
- src/cc/geosrv/Mercator.java

7.14 cc.ws.ReplayMgr Class Reference

Inheritance diagram for cc.ws.ReplayMgr:



Collaboration diagram for cc.ws.ReplayMgr:



Classes

- class Storm

Public Member Functions

- ReplayMgr ()
- void init ()
- void run ()
- void doPost (HttpServletRequest oReq, HttpServletResponse oRep) throws IOException

Protected Attributes

- String m_sStormFile

Private Attributes

- final ArrayList< Storm > m_oStorms = new ArrayList()

7.14.1 Detailed Description

Definition at line 15 of file ReplayMgr.java.

7.14.2 Constructor & Destructor Documentation

7.14.2.1 ReplayMgr()

```
cc.ws.ReplayMgr.ReplayMgr ( )  [inline]
```

Definition at line 21 of file ReplayMgr.java.

```
00022     {
00023     }
```

7.14.3 Member Function Documentation

7.14.3.1 doPost()

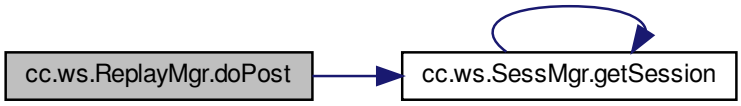
```
void cc.ws.ReplayMgr.doPost (
            HttpServletRequest oReq,
            HttpServletResponse oRep ) throws IOException  [inline]
```

Definition at line 89 of file ReplayMgr.java.

```
00091     {
00092         Session oSess = SessMgr.getSession(oReq);
00093         if (oSess == null)
00094         {
00095             oRep.sendError(401);
00096             return;
00097         }
00098 
00099         try (JsonGenerator oJson = Json.createGenerator(oRep.getOutputStream()))
00100         {
00101             oJson.writeStartArray(); // start outer JSON array
00102             for (Storm oStorm : m_oStorms)
00103             {
00104                 oJson.writeStartArray(); // start record
00105                 oJson.write(oStorm.m_sStart).write(oStorm.m_sEnd).write(oStorm.m_sHours).
00106                     write(oStorm.m_nAvg).write(oStorm.m_nMax).write("");
00107                 oJson.writeEnd(); // end record
00108             }
00109             oJson.writeEnd(); // end outer JSON array
00110         }
00111     }
```

References cc.ws.SessMgr.getSession(), and cc.ws.ReplayMgr.m_oStorms.

Here is the call graph for this function:



7.14.3.2 init()

```
void cc.ws.ReplayMgr.init ( )  [inline]
```

Definition at line 27 of file ReplayMgr.java.

```
00028     {
00029         String sStormFile = getServletConfig().getInitParameter("stormfile");
00030         if (sStormFile != null && sStormFile.length() > 0)
00031         {
00032             m_sStormFile = sStormFile;
00033             new Thread(this).start();
00034         }
00035     }
```

References cc.ws.ReplayMgr.m_sStormFile.

7.14.3.3 run()

```
void cc.ws.ReplayMgr.run ( )  [inline]
```

Definition at line 39 of file ReplayMgr.java.

```
00040     {
00041         int nCells = Integer.MIN_VALUE;
00042         try (CsvReader oIn = new CsvReader(new FileInputStream(m_sStormFile)))
00043         {
00044             SimpleDateFormat oFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm");
00045             StringBuilder sBuf = new StringBuilder();
00046 
00047             oIn.readLine(); // skip header: start end avg max min
00048             while (oIn.readLine() > 0)
00049             {
00050                 oIn.parseString(sBuf, 0); // get start date as string
00051                 String sStart = sBuf.toString();
00052 
00053                 oIn.parseString(sBuf, 1); // get end date as string
00054                 String sEnd = sBuf.toString();
00055 
00056                 double dDur = oFormat.parse(sEnd).getTime() - oFormat.parse(sStart).getTime();
00057 
00058                 Storm oStorm = new Storm();
00059                 oStorm.m_sStart = sStart;
00060                 oStorm.m_sEnd = sEnd;
00061                 oStorm.m_sHours = String.format("%03.1f", dDur / 3600000.0);
00062                 oStorm.m_nAvg = (int)Math.round(oIn.parseDouble(2));
00063                 oStorm.m_nMax = oIn.parseInt(3);
00064                 oStorm.m_nMin = oIn.parseInt(4);
00065                 m_oStorms.add(oStorm);
00066 
00067                 if (oStorm.m_nMax > nCells)
00068                     nCells = oStorm.m_nMax; // find greatest cell count
00069             }
00070         }
00071         catch (Exception oEx)
00072         {
00073             oEx.printStackTrace();
00074         }
00075 
00076         if (nCells <= 0)
00077             return;
00078 
00079         for (Storm oStorm : m_oStorms)
00080         {
00081             oStorm.m_nAvg = 100 * oStorm.m_nAvg / nCells;
00082             oStorm.m_nMin = 100 * oStorm.m_nMin / nCells;
00083             oStorm.m_nMax = 100 * oStorm.m_nMax / nCells;
00084         }
00085     }
```

References cc.ws.ReplayMgr.Storm.m_nMax, cc.ws.ReplayMgr.m_oStorms, and cc.ws.ReplayMgr.m_sStormFile.

7.14.4 Member Data Documentation

7.14.4.1 m_oStorms

```
final ArrayList<Storm> cc.ws.ReplayMgr.m_oStorms = new ArrayList()  [private]
```

Definition at line 18 of file ReplayMgr.java.

Referenced by cc.ws.ReplayMgr.doPost(), and cc.ws.ReplayMgr.run().

7.14.4.2 m_sStormFile

```
String cc.ws.ReplayMgr.m_sStormFile  [protected]
```

Definition at line 17 of file ReplayMgr.java.

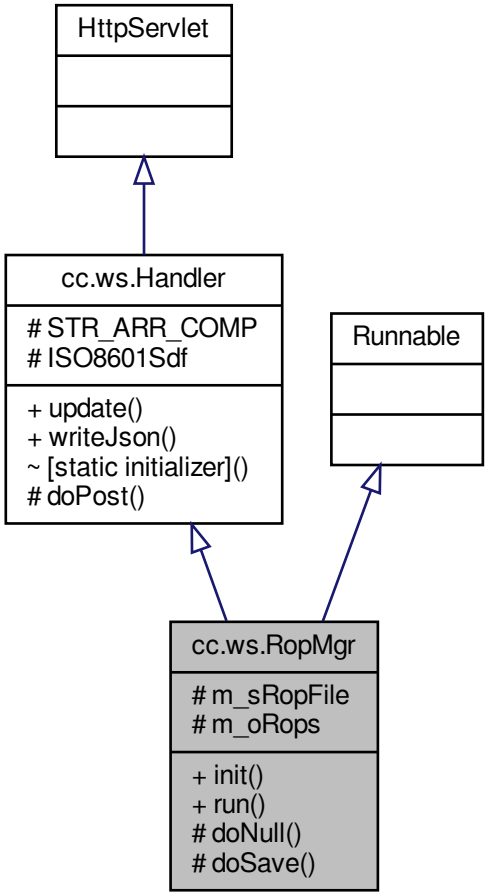
Referenced by cc.ws.ReplayMgr.init(), and cc.ws.ReplayMgr.run().

The documentation for this class was generated from the following file:

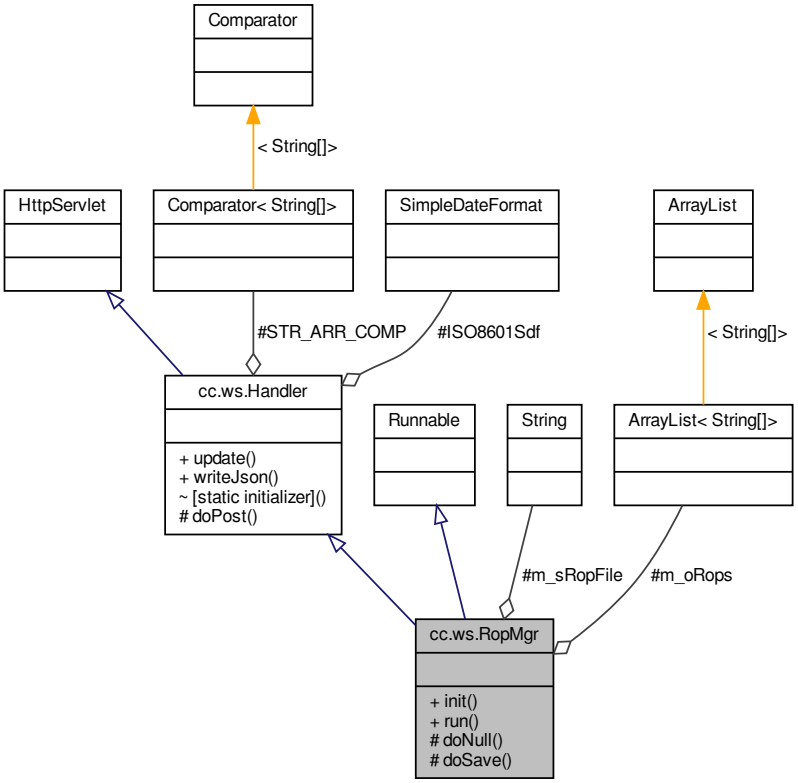
- src/cc/ws/ReplayMgr.java

7.15 cc.ws.RopMgr Class Reference

Inheritance diagram for cc.ws.RopMgr:



Collaboration diagram for cc.ws.RopMgr:



Public Member Functions

- void init ()
- void run ()

Protected Member Functions

- void doNull (Session oSess, HttpServletRequest oReq, PrintWriter oOut)
- void doSave (Session oSess, HttpServletRequest oReq, PrintWriter oOut) throws IOException

Protected Attributes

- String m_sRopFile
- ArrayList< String[ ]> m_oRops = new ArrayList()

Additional Inherited Members

7.15.1 Detailed Description

Author

Federal Highway Administration

Definition at line 23 of file RopMgr.java.

7.15.2 Member Function Documentation

7.15.2.1 doNull()

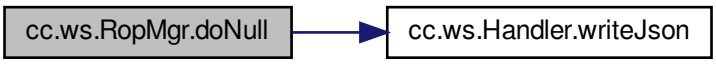
```
void cc.ws.RopMgr.doNull (
            Session oSess,
            HttpServletRequest oReq,
            PrintWriter oOut )  [inline], [protected]
```

Definition at line 69 of file RopMgr.java.

```
00070     {
00071         oOut.write("{");
00072         synchronized (m_oRops)
00073         {
00074             int nSize = m_oRops.size();
00075             if (nSize > 0)
00076             {
00077                 String[] sRop = m_oRops.get(0);
00078                 writeJson(oOut, sRop[0], sRop[sRop.length - 1]);
00079                 for (int i = 1; i < nSize; i++)
00080                 {
00081                     sRop = m_oRops.get(i);
00082                     oOut.write(",");
00083                     writeJson(oOut, sRop[0], sRop[sRop.length - 1]); // {"uuid" : JSON, "uuid" : JSON,
       ...}
00084                 }
00085             }
00086         }
00087         oOut.write("}");
00088     }
```

References cc.ws.RopMgr.m_oRops, and cc.ws.Handler.writeJson().

Here is the call graph for this function:



7.15.2.2 doSave()

```
void cc.ws.RopMgr.doSave (
            Session oSess,
            HttpServletRequest oReq,
            PrintWriter oOut ) throws IOException  [inline], [protected]
```

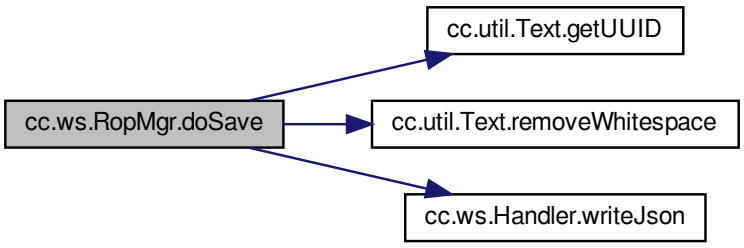
Definition at line 91 of file RopMgr.java.

```
00093     {
00094         String[] sRop;
00095         synchronized (m_oRops)
00096         {
```

```
00097             String[] sSearch = new String[1];
00098             sSearch[0] = oReq.getParameter("id");
00099             if (sSearch[0] == null || sSearch[0].length() == 0) // no id so create a new one
00100                 sSearch[0] = Text.getUUID();
00101 
00102             StringBuilder sBuffer = new StringBuilder(oReq.getParameter("data"));
00103             if (sBuffer.indexOf("\"status\"") < 0) // add default status of unused to json
00104             {
00105                 Text.removeWhitespace(sBuffer);
00106                 sBuffer.insert(sBuffer.length() - 2, ",\"status\":\"U\"");
00107             }
00108 
00109             int nIndex = Collections.binarySearch(m_oRops, sSearch, STR_ARR_COMP);
00110             if (nIndex < 0)
00111             {
00112                 nIndex = ~nIndex;
00113                 m_oRops.add(nIndex, new String[]{sSearch[0], oSess.m_oUser.m_sUser, null,
      sBuffer.toString()}); // create new array
00114             }
00115             sRop = m_oRops.get(nIndex);
00116             synchronized (ISO8601Sdf)
00117             {
00118                 sRop[2] = ISO8601Sdf.format(System.currentTimeMillis());
00119             }
00120 
00121             if (sRop[3].contains("\"status\":\"D\"")) // remove from list if the status is deleted
00122                 m_oRops.remove(nIndex);
00123 
00124             try (BufferedWriter oFileOut = new BufferedWriter(new FileWriter(m_sRopFile, true)))
00125             {
00126                 oFileOut.write(sRop[0]);
00127                 for (int i = 1; i < sRop.length; i++)
00128                 {
00129                     oFileOut.write("\t");
00130                     oFileOut.write(sRop[i]);
00131                 }
00132                 oFileOut.write("\n");
00133             }
00134         }
00135         oOut.write("{");
00136         writeJson(oOut, sRop[0], sRop[sRop.length - 1]); // id:data
00137         oOut.write("}");
00138     }
```

References cc.util.Text.getUUID(), cc.ws.Handler.ISO8601Sdf, cc.ws.RopMgr.m_oRops, cc.ws.RopMgr.m_sRopFile, cc.util.Text.removeWhitespace(), cc.ws.Handler.STR_ARR_COMP, and cc.ws.Handler.writeJson().

Here is the call graph for this function:



7.15.2.3 init()

```
void cc.ws.RopMgr.init ( )  [inline]
```

Definition at line 30 of file RopMgr.java.

```
00031     {
00032         String sRopFile = getServletConfig().getInitParameter("ropfile");
00033         if (sRopFile != null && sRopFile.length() > 0)
00034         {
00035             m_sRopFile = sRopFile;
00036             new Thread(this).start();
00037         }
00038     }
```

References cc.ws.RopMgr.m_sRopFile.

7.15.2.4 run()

```
void cc.ws.RopMgr.run ( )  [inline]
```

Definition at line 42 of file RopMgr.java.

```
00043     {
00044         try (CsvReader oIn = new CsvReader(new FileInputStream(m_sRopFile), '\t'))
00045         {
00046             int nCols;
00047             String[] sSearch = new String[1];
00048             synchronized(m_oRops)
00049             {
00050                 while ((nCols = oIn.readLine()) > 0)
00051                 {
00052                     sSearch[0] = oIn.parseString(0);
00053                     int nIndex = Collections.binarySearch(m_oRops, sSearch, STR_ARR_COMP); // search
      for id in list
00054                     if (nIndex < 0)
00055                     {
00056                         nIndex = ~nIndex;
00057                         m_oRops.add(nIndex, new String[nCols]); // right now we have
      uuid,user,timestamp,status,JSON
00058                     }
00059                     update(m_oRops.get(nIndex), nCols, oIn); // replace with most recent update
00060                 }
00061             }
00062         }
00063         catch (Exception oEx)
00064         {
00065         }
00066     }
```

References cc.ws.RopMgr.m_oRops, cc.ws.RopMgr.m_sRopFile, cc.ws.Handler.STR_ARR_COMP, and cc.ws.Handler.update().

Here is the call graph for this function:

7.15.3 Member Data Documentation

7.15.3.1 m_oRops

```
ArrayList<String[]> cc.ws.RopMgr.m_oRops = new ArrayList()  [protected]
```

Definition at line 26 of file RopMgr.java.

Referenced by cc.ws.RopMgr.doNull(), cc.ws.RopMgr.doSave(), and cc.ws.RopMgr.run().

7.15.3.2 m_sRopFile

```
String cc.ws.RopMgr.m_sRopFile  [protected]
```

Definition at line 25 of file RopMgr.java.

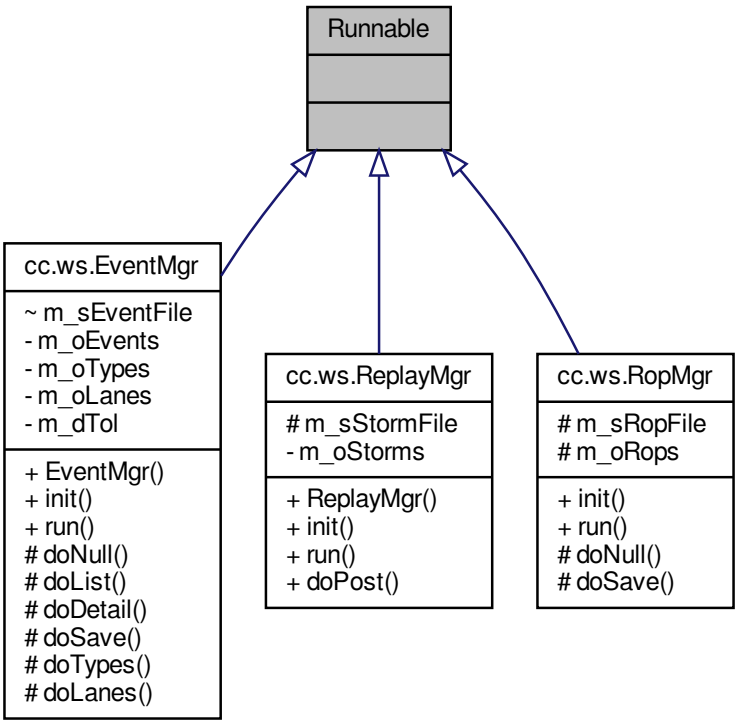
Referenced by cc.ws.RopMgr.doSave(), cc.ws.RopMgr.init(), and cc.ws.RopMgr.run().

The documentation for this class was generated from the following file:

- src/cc/ws/RopMgr.java

7.16 Runnable Class Reference

Inheritance diagram for Runnable:



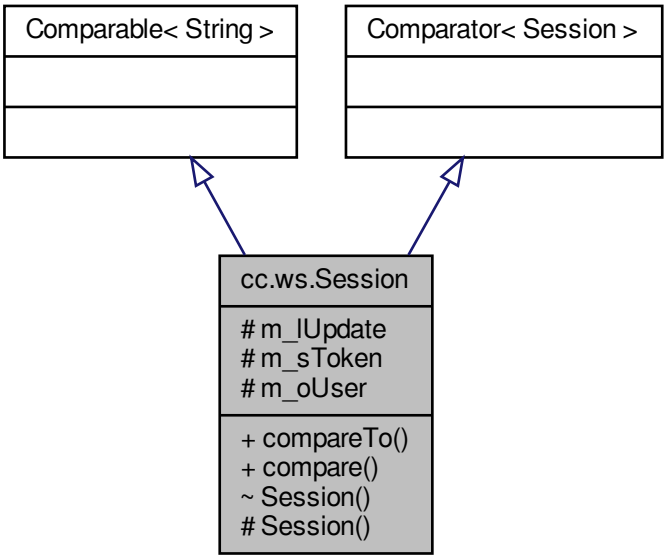
Collaboration diagram for Runnable:

The documentation for this class was generated from the following file:

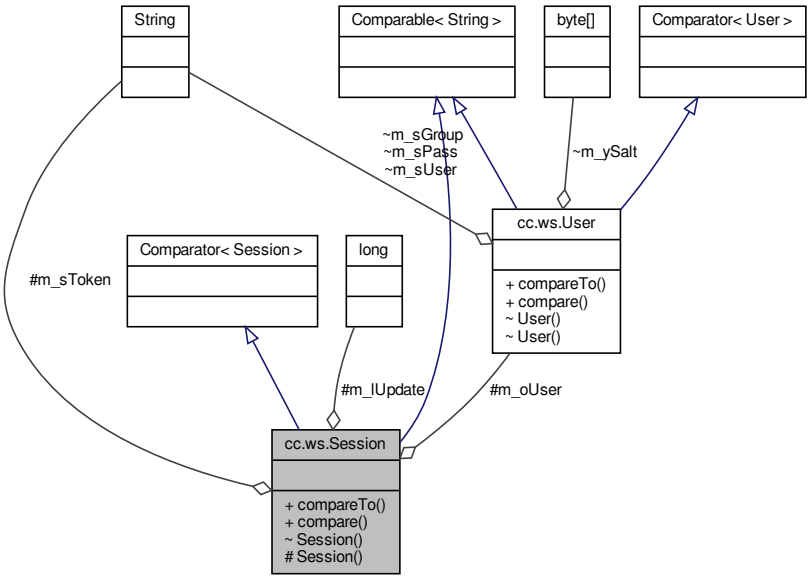
- src/cc/ws/EventMgr.java

7.17 cc.ws.Session Class Reference

Inheritance diagram for cc.ws.Session:



Collaboration diagram for cc.ws.Session:



Public Member Functions

- int compareTo (String sKey)
- int compare (Session oLhs, Session oRhs)

Protected Member Functions

- Session ()

Protected Attributes

- long m_lUpdate
- String m_sToken
- User m_oUser

Package Functions

- Session (String sKey)

7.17.1 Detailed Description

Definition at line 6 of file Session.java.

7.17.2 Constructor & Destructor Documentation

7.17.2.1 Session() [1/2]

```
cc.ws.Session.Session ( )  [inline], [protected]
```

Definition at line 13 of file Session.java.

```
00014     {
00015     }
```

7.17.2.2 Session() [2/2]

```
cc.ws.Session.Session (
            String sKey )  [inline], [package]
```

Definition at line 18 of file Session.java.

```
00019     {
00020         m_sToken = sKey;
00021     }
```

References cc.ws.Session.m_sToken.

7.17.3 Member Function Documentation

7.17.3.1 compare()

```
int cc.ws.Session.compare (
            Session oLhs,
            Session oRhs )  [inline]
```

Definition at line 32 of file Session.java.

```
00033     {
00034         return oLhs.m_sToken.compareTo(oRhs.m_sToken);
00035     }
```

References cc.ws.Session.m_sToken.

7.17.3.2 compareTo()

```
int cc.ws.Session.compareTo (
            String sKey )  [inline]
```

Definition at line 25 of file Session.java.

```
00026     {
00027         return m_sToken.compareTo(sKey);
00028     }
```

References cc.ws.Session.m_sToken.

7.17.4 Member Data Documentation

7.17.4.1 m_lUpdate

```
long cc.ws.Session.m_lUpdate  [protected]
```

Definition at line 8 of file Session.java.

Referenced by cc.ws.SessMgr.getSession().

7.17.4.2 m_oUser

```
User cc.ws.Session.m_oUser  [protected]
```

Definition at line 10 of file Session.java.

7.17.4.3 m_sToken

```
String cc.ws.Session.m_sToken  [protected]
```

Definition at line 9 of file Session.java.

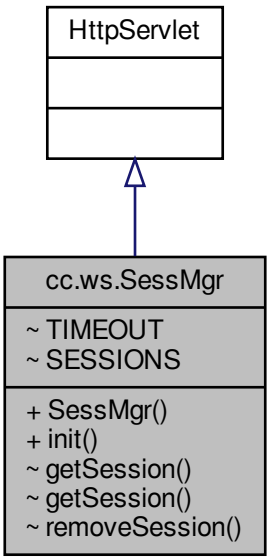
Referenced by cc.ws.Session.Session(), cc.ws.Session.compare(), cc.ws.Session.compareTo(), and cc.ws.UserMgr.doPost().

The documentation for this class was generated from the following file:

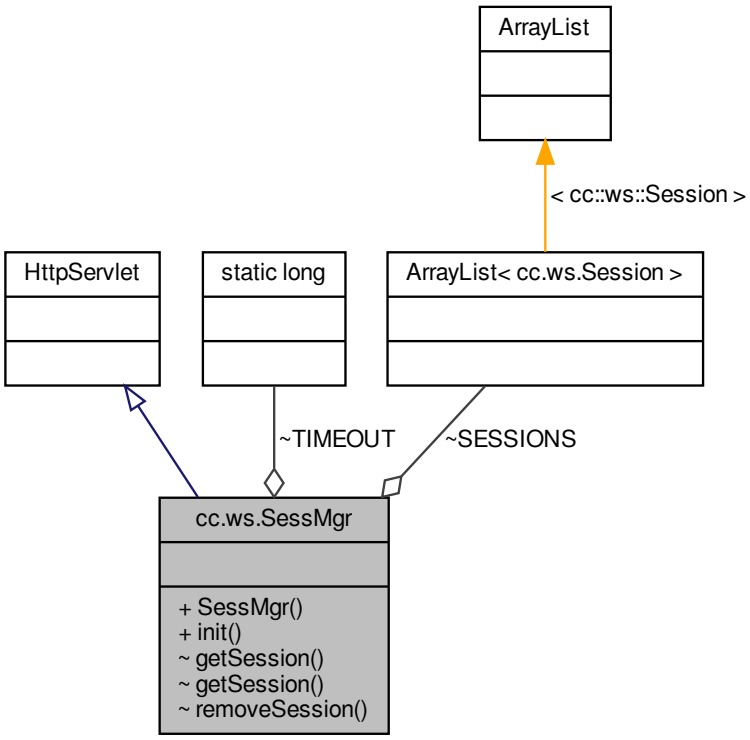
- src/cc/ws/Session.java

7.18 cc.ws.SessMgr Class Reference

Inheritance diagram for cc.ws.SessMgr:



Collaboration diagram for cc.ws.SessMgr:



Public Member Functions

- SessMgr ()
- void init ()

Static Package Functions

- static Session getSession (HttpServletRequest oReq)
- static Session getSession (HttpServletRequest oReq, boolean bCreate)
- static void removeSession (Session oSess)

Static Package Attributes

- static long TIMEOUT = 1800000
- static final ArrayList< Session > SESSIONS = new ArrayList()

7.18.1 Detailed Description

Definition at line 12 of file SessMgr.java.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 SessMgr()

```
cc.ws.SessMgr.SessMgr ( )  [inline]
```

Definition at line 18 of file SessMgr.java.

```
00019     {
00020     }
```

7.18.3 Member Function Documentation

7.18.3.1 getSession() [1/2]

```
static Session cc.ws.SessMgr.getSession (
            HttpServletRequest oReq )  [inline], [static], [package]
```

Definition at line 33 of file SessMgr.java.

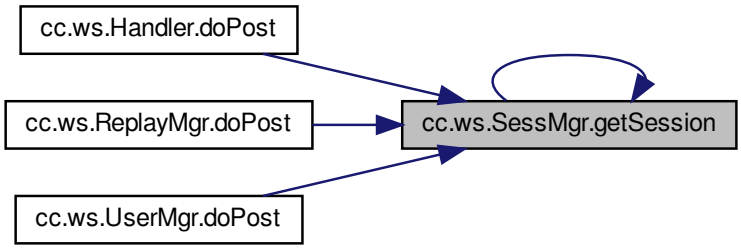
```
00034     {
00035         return getSession(oReq, false);
00036     }
```

References cc.ws.SessMgr.getSession().

Referenced by cc.ws.Handler.doPost(), cc.ws.ReplayMgr.doPost(), cc.ws.UserMgr.doPost(), and cc.ws.SessMgr.getSession().

Here is the call graph for this function:

Here is the caller graph for this function:



7.18.3.2 getSession() [2/2]

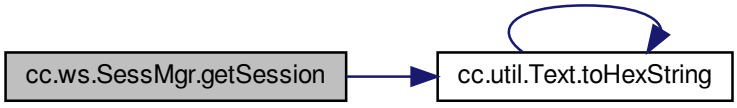
```
static Session cc.ws.SessMgr.getSession (
            HttpServletRequest oReq,
            boolean bCreate )  [inline], [static], [package]
```

Definition at line 39 of file SessMgr.java.

```
00040     {
00041         String sToken = oReq.getParameter("token");
00042         if (sToken == null)
00043             sToken = "";
00044 
00045         Session oSess = null;
00046         synchronized(SESSIONS)
00047         {
00048             int nIndex = Collections.binarySearch(SESSIONS, sToken);
00049             if (nIndex >= 0)
00050             {
00051                 oSess = SESSIONS.get(nIndex);
00052                 if (oSess.m_lUpdate < System.currentTimeMillis())
00053                 {
00054                     SESSIONS.remove(nIndex);
00055                     return null;
00056                 }
00057             }
00058             else if (bCreate)
00059             {
00060                 byte[] yBytes = new byte[16];
00061                 try
00062                 {
00063                     SecureRandom oRng = SecureRandom.getInstance("SHA1PRNG");
00064                     do
00065                     {
00066                         oRng.nextBytes(yBytes); // ensure no duplicates
00067                         oSess = new Session(Text.toHexString(yBytes));
00068                         nIndex = Collections.binarySearch(SESSIONS, oSess, oSess);
00069                     }
00070                     while (nIndex >= 0);
00071                     SESSIONS.add(~nIndex, oSess); // save new session
00072                 }
00073                 catch (Exception oEx)
00074                 {
00075                 }
00076             }
00077         }
00078 
00079         if (oSess != null)
00080             oSess.m_lUpdate = System.currentTimeMillis() + TIMEOUT;
00081 
00082         return oSess;
00083     }
```

References cc.ws.Session.m_lUpdate, cc.ws.SessMgr.SESSIONS, cc.ws.SessMgr.TIMEOUT, and cc.util.Text.toHexString().

Here is the call graph for this function:



7.18.3.3 init()

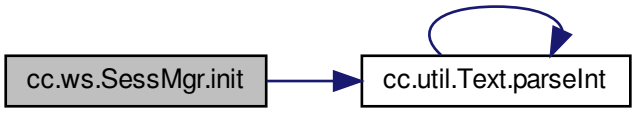
```
void cc.ws.SessMgr.init ( )  [inline]
```

Definition at line 24 of file SessMgr.java.

```
00025     {
00026         ServletConfig oConf = getServletConfig();
00027         String sExp = oConf.getInitParameter("timeout");
00028         if (sExp != null)
00029             TIMEOUT = Text.parseInt(sExp);
00030     }
```

References cc.util.Text.parseInt(), and cc.ws.SessMgr.TIMEOUT.

Here is the call graph for this function:



7.18.3.4 removeSession()

```
static void cc.ws.SessMgr.removeSession (
             Session oSess )  [inline], [static], [package]
```

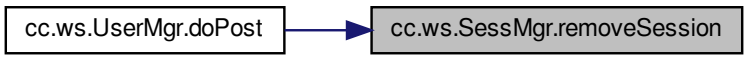
Definition at line 86 of file SessMgr.java.

```
00087     {
00088         if (oSess == null)
00089             return;
00090 
00091         synchronized(SESSIONS)
00092         {
00093             int nIndex = Collections.binarySearch(SESSIONS, oSess, oSess);
00094             if (nIndex >= 0)
00095                 SESSIONS.remove(nIndex);
00096         }
00097     }
```

References cc.ws.SessMgr.SESSIONS.

Referenced by cc.ws.UserMgr.doPost().

Here is the caller graph for this function:



7.18.4 Member Data Documentation

7.18.4.1 SESSIONS

```
final ArrayList<Session> cc.ws.SessMgr.SESSIONS = new ArrayList()  [static], [package]
```

Definition at line 15 of file SessMgr.java.

Referenced by cc.ws.SessMgr.getSession(), and cc.ws.SessMgr.removeSession().

7.18.4.2 TIMEOUT

```
long cc.ws.SessMgr.TIMEOUT = 1800000  [static], [package]
```

Definition at line 14 of file SessMgr.java.

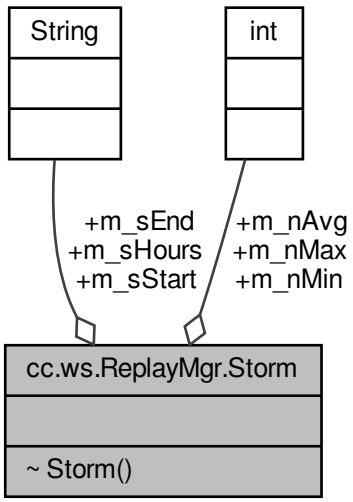
Referenced by cc.ws.SessMgr.getSession(), and cc.ws.SessMgr.init().

The documentation for this class was generated from the following file:

- src/cc/ws/SessMgr.java

7.19 cc.ws.ReplayMgr.Storm Class Reference

Collaboration diagram for cc.ws.ReplayMgr.Storm:



Public Attributes

- String m_sStart
- String m_sEnd
- String m_sHours
- int m_nMin
- int m_nMax
- int m_nAvg

Package Functions

- Storm ()

7.19.1 Detailed Description

Definition at line 114 of file ReplayMgr.java.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 Storm()

```
cc.ws.ReplayMgr.Storm.Storm ( )  [inline], [package]
```

Definition at line 124 of file ReplayMgr.java.

```
00125         {
00126         }
```

7.19.3 Member Data Documentation

7.19.3.1 m_nAvg

```
int cc.ws.ReplayMgr.Storm.m_nAvg
```

Definition at line 121 of file ReplayMgr.java.

7.19.3.2 m_nMax

```
int cc.ws.ReplayMgr.Storm.m_nMax
```

Definition at line 120 of file ReplayMgr.java.

Referenced by cc.ws.ReplayMgr.run().

7.19.3.3 m_nMin

```
int cc.ws.ReplayMgr.Storm.m_nMin
```

Definition at line 119 of file ReplayMgr.java.

7.19.3.4 m_sEnd

```
String cc.ws.ReplayMgr.Storm.m_sEnd
```

Definition at line 117 of file ReplayMgr.java.

7.19.3.5 m_sHours

```
String cc.ws.ReplayMgr.Storm.m_sHours
```

Definition at line 118 of file ReplayMgr.java.

7.19.3.6 m_sStart

```
String cc.ws.ReplayMgr.Storm.m_sStart
```

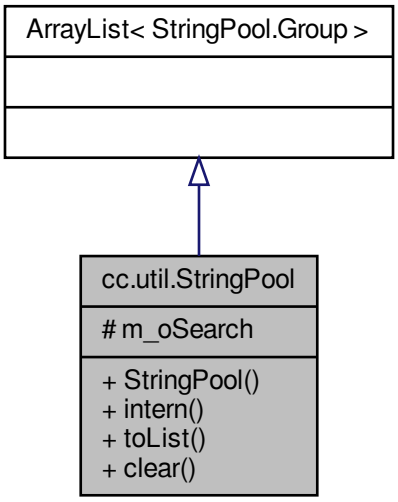
Definition at line 116 of file ReplayMgr.java.

The documentation for this class was generated from the following file:

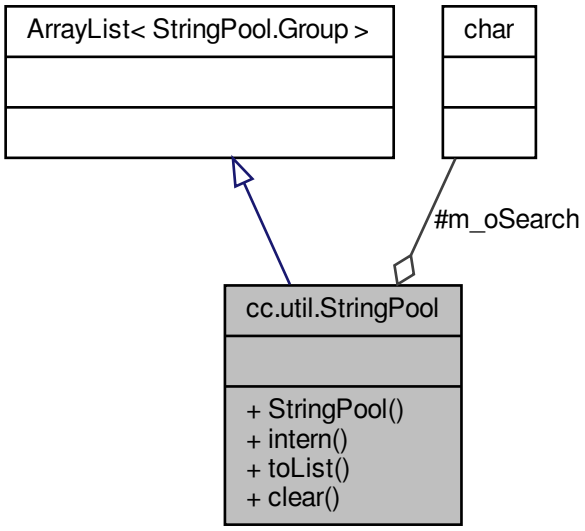
- src/cc/ws/ReplayMgr.java

7.20 cc.util.StringPool Class Reference

Inheritance diagram for cc.util.StringPool:



Collaboration diagram for cc.util.StringPool:



Classes

- class Group

Public Member Functions

- StringPool ()
- String intern (String sVal)
- ArrayList< String > toList ()
- void clear ()

Protected Attributes

- char[ ] m_oSearch = new char[2]

7.20.1 Detailed Description

Definition at line 7 of file StringPool.java.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 StringPool()

```
cc.util.StringPool.StringPool ( )  [inline]
```

Definition at line 12 of file StringPool.java.

```
00013     {
00014     }
```

7.20.3 Member Function Documentation

7.20.3.1 clear()

```
void cc.util.StringPool.clear ( )  [inline]
```

Definition at line 63 of file StringPool.java.

```
00064     {
00065         for (Group oGroup : this)
00066             oGroup.clear();
00067 
00068         super.clear();
00069     }
```

7.20.3.2 intern()

```
String cc.util.StringPool.intern (
            String sVal )  [inline]
```

Definition at line 17 of file StringPool.java.

```
00018     {
00019         int nIndex = m_oSearch.length;
00020         while (nIndex-- > 0) // create search key
00021         {
00022             if (nIndex < sVal.length())
00023                 m_oSearch[nIndex] = Character.toUpperCase(sVal.charAt(nIndex));
00024             else
00025                 m_oSearch[nIndex] = 0;
00026         }
00027
00028         nIndex = Collections.binarySearch(this, m_oSearch);
00029         if (nIndex < 0) // completely new string group array
00030         {
00031             nIndex = ~nIndex;
00032             Group oGroup = new Group(m_oSearch);
00033             add(nIndex, oGroup);
00034         }
00035
00036         Group oGroup = get(nIndex);
00037         nIndex = Collections.binarySearch(oGroup, sVal);
00038         if (nIndex < 0)
00039         {
00040             oGroup.add(~nIndex, sVal);
00041             return sVal;
00042         }
00043         return oGroup.get(nIndex);
00044     }
```

References cc.util.StringPool.m_oSearch.

7.20.3.3 toList()

```
ArrayList< String > cc.util.StringPool.toList ( )  [inline]
```

Definition at line 47 of file StringPool.java.

```
00048     {
00049         int nSize = 0; // determine space requirements
00050         for (Group oGroup : this)
00051             nSize += oGroup.size();
00052
00053         ArrayList<String> oList = new ArrayList(nSize);
00054         for (Group oGroup : this)
00055             oList.addAll(oGroup);
00056
00057         Collections.sort(oList); // correct pool grouping order
00058         return oList;
00059     }
```

7.20.4 Member Data Documentation

7.20.4.1 m_oSearch

```
char [] cc.util.StringPool.m_oSearch = new char[2]  [protected]
```

Definition at line 9 of file StringPool.java.

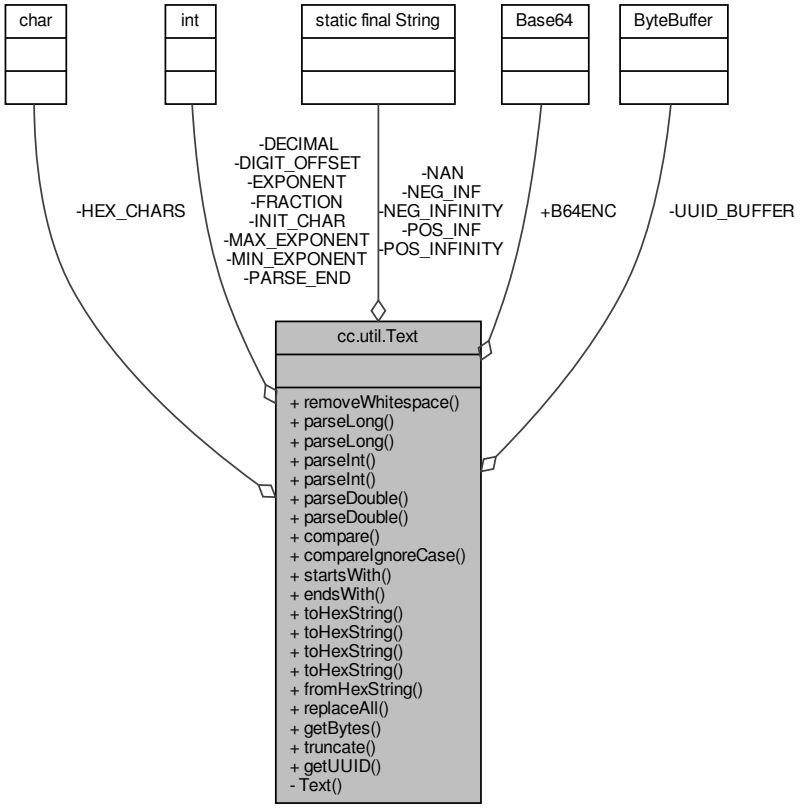
Referenced by cc.util.StringPool.intern().

The documentation for this class was generated from the following file:

- src/cc/util/StringPool.java

7.21 cc.util.Text Class Reference

Collaboration diagram for cc.util.Text:



Static Public Member Functions

- static void removeWhitespace (StringBuilder sBuffer)
- static long parseLong (CharSequence iCharSeq)
- static long parseLong (CharSequence iCharSeq, int nPos, int nEndPos) throws NumberFormatException
- static int parseInt (CharSequence iCharSeq)
- static int parseInt (CharSequence iCharSeq, int nPos, int nEndPos) throws NumberFormatException
- static double parseDouble (CharSequence iCharSeq, int nPos, int nEndPos) throws IndexOutOfBounds↩ Exception, NumberFormatException
- static double parseDouble (CharSequence iCharSeq)
- static int compare (CharSequence iSeqL, CharSequence iSeqR)
- static int compareIgnoreCase (CharSequence iSeqL, CharSequence iSeqR)
- static boolean startsWith (CharSequence iSource, CharSequence iPrefix)
- static boolean endsWith (CharSequence iSource, CharSequence iSuffix)
- static String toHexString (byte[ ] yBytes)
- static void toHexString (byte[ ] yBytes, StringBuilder sBuf)
- static String toHexString (byte[ ] yBytes, int nOffset, int nLength)
- static void toHexString (byte[ ] yBytes, int nOffset, int nLength, StringBuilder sBuf)
- static byte[ ] fromHexString (StringBuilder sBuf)
- static void replaceAll (StringBuilder sBuffer, String sSearch, String sReplace)
- static int getBytes (byte[ ] yBuffer, CharSequence iCharSeq)
- static String truncate (String sValue, int nLength)
- static String getUUID ()

Static Public Attributes

- static final Base64.Encoder B64ENC = Base64.getUrlEncoder().withoutPadding()

Private Member Functions

- Text ()

Static Private Attributes

- static final char[ ] HEX_CHARS
- static final int DIGIT_OFFSET = 48
- static final int MIN_EXPONENT = -323
- static final int MAX_EXPONENT = 308
- static final int INIT_CHAR = 0
- static final int DECIMAL = 1
- static final int FRACTION = 2
- static final int EXPONENT = 3
- static final int PARSE_END = 4
- static final String POS_INF = "\u221E"
- static final String NEG_INF = "-\u221E"
- static final String POS_INFINITY = "Infinity"
- static final String NEG_INFINITY = "-Infinity"
- static final String NAN = "NaN"
- static final ByteBuffer UUID_BUFFER = ByteBuffer.allocate(16)

7.21.1 Detailed Description

Provides methods to parse strings to extract numerical values. Also contains methods to format and compare character sequences.

Definition at line 12 of file Text.java.

7.21.2 Constructor & Destructor Documentation

7.21.2.1 Text()

`cc.util.Text.Text ( )  [inline], [private]`

Default Constructor

Creates new instances of `Text`

Definition at line 93 of file Text.java.

```
00094         {
00095     }
```

7.21.3 Member Function Documentation

7.21.3.1 compare()

```
static int cc.util.Text.compare (
            CharSequence iSeqL,
            CharSequence iSeqR )  [inline], [static]
```

Lexicographically compare two character sequences. Using the character sequence interface enables the mixing of comparisons between `String`, `StringBuffer`, and `StringBuilder` objects. The character values at each index of the sequences is compared up to the minimum number of available characters. The sequence lengths determine the comparison when the contents otherwise appear to be equal.

Parameters

iSeqL	the first character sequence to be compared
iSeqR	the second character sequence to be compared

Returns

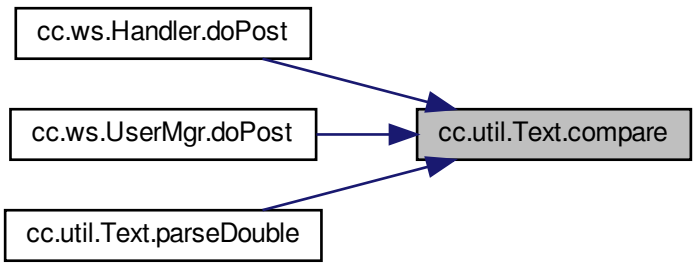
a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second

Definition at line 425 of file Text.java.

```
00425                                                                   {
00426         int nCompare = 0;
00427         int nIndex = -1;
00428         int nLimit = Math.min(iSeqL.length(), iSeqR.length());
00429
00430         while (nCompare == 0 && ++nIndex < nLimit)
00431             nCompare = (iSeqL.charAt(nIndex) - iSeqR.charAt(nIndex));
00432
00433         if (nCompare == 0)
00434             nCompare = (iSeqL.length() - iSeqR.length());
00435
00436         return nCompare;
00437     }
```

Referenced by cc.ws.Handler.doPost(), cc.ws.UserMgr.doPost(), and cc.util.Text.parseDouble().

Here is the caller graph for this function:



7.21.3.2 compareIgnoreCase()

```
static int cc.util.Text.compareIgnoreCase (
            CharSequence iSeqL,
            CharSequence iSeqR )  [inline], [static]
```

Lexicographically compare two character sequences. Comparison is performed without regard to differing character case.

Parameters

iSeqL	the first character sequence to be compared
iSeqR	the second character sequence to be compared

Returns

a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second

Definition at line 449 of file Text.java.

```
00449                                                                       {
00450         int nCompare = 0;
00451         int nIndex = -1;
00452         int nLimit = Math.min(iSeqL.length(), iSeqR.length());
00453
00454         while (nCompare == 0 && ++nIndex < nLimit)
00455             nCompare =
00456                     (
00457                             Character.toLowerCase(iSeqL.charAt(nIndex)) -
00458                                     Character.toLowerCase(iSeqR.charAt(nIndex))
00459                     );
00460
00461         if (nCompare == 0)
00462             nCompare = (iSeqL.length() - iSeqR.length());
00463
00464         return nCompare;
00465     }
```

7.21.3.3 endsWith()

```
static boolean cc.util.Text.endsWith (
            CharSequence iSource,
            CharSequence iSuffix )  [inline], [static]
```

Tests if the source character sequence ends with the suffix character sequence.

Parameters

iSource	the character sequence to check
iSuffix	the search character sequence

Returns

`true` if the characters at the end of the source match the characters in the suffix. `false` otherwise.

Definition at line 502 of file Text.java.

```
00502                                                                       {
00503         int nIndex = iSuffix.length();
00504         int nSrcIndex = iSource.length();
00505
00506         // the source cannot end with a pattern with more characters
00507         if (nIndex > nSrcIndex)
00508             return false;
00509
00510         // compare each characters starting at the end of the sequence
00511         boolean bMatch = true;
00512         while (bMatch && nIndex-- > 0)
00513             bMatch = (iSource.charAt(--nSrcIndex) == iSuffix.charAt(nIndex));
00514
00515         return bMatch;
00516     }
```

7.21.3.4 fromHexString()

```
static byte[] cc.util.Text.fromHexString (
             StringBuilder sBuf )  [inline], [static]
```

Converts a hexadecimal sequence into a byte array

Parameters

sBuf	String buffer holding hexadecimal characters.

Returns

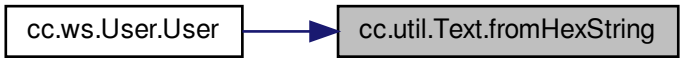
A byte array containing the interpreted bytes.

Definition at line 584 of file Text.java.

```
00585         {
00586             if (sBuf == null || sBuf.length() == 0)
00587                 return null;
00588
00589             if (sBuf.length() % 2 != 0)
00590                 sBuf.append("0");
00591
00592             byte[] yBytes = new byte[sBuf.length() / 2];
00593             for (int nIndex = 0; nIndex < yBytes.length; nIndex++)
00594             {
00595                 int nPos = nIndex * 2;
00596                 yBytes[nIndex] = (byte)((Character.digit(sBuf.charAt(nPos), 16) << 4) +
00597                     Character.digit(sBuf.charAt(nPos + 1), 16));
00598             }
00599             return yBytes;
00600     }
```

Referenced by cc.ws.User.User().

Here is the caller graph for this function:



7.21.3.5 getBytes()

```
static int cc.util.Text.getBytes (
            byte[] yBuffer,
            CharSequence iCharSeq )  [inline], [static]
```

Copies the contents of a character sequence into the provided byte buffer. No locale translation is performed. This is mostly useful for UTF-8 and ASCII encoded strings.

Parameters

yBuffer	The byte buffer where characters are copied.
iCharSeq	The sequence of characters to convert to bytes.

Returns

The number of bytes copied into the buffer.

Definition at line 631 of file Text.java.

```
00631                                                                   {
00632         // determine the maximum the available capacity
00633         int nLength = iCharSeq.length();
00634         if (nLength > yBuffer.length)
00635             nLength = yBuffer.length;
00636
00637         for (int nIndex = 0; nIndex < nLength; nIndex++)
00638             yBuffer[nIndex] = (byte) iCharSeq.charAt(nIndex);
00639
00640         return nLength;
00641     }
```

7.21.3.6 getUUID()

```
static String cc.util.Text.getUUID ( )  [inline], [static]
```

Definition at line 660 of file Text.java.

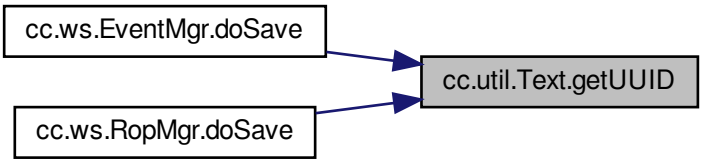
```
00661     {
00662         UUID oUuid = UUID.randomUUID();
00663         synchronized(UUID_BUFFER) // shared byte buffer
00664         {
```

```
00665             UUID_BUFFER.clear();
00666             UUID_BUFFER.putLong(oUuid.getMostSignificantBits());
00667             UUID_BUFFER.putLong(oUuid.getLeastSignificantBits());
00668             return B64ENC.encodeToString(UUID_BUFFER.array());
00669         }
00670     }
```

References cc.util.Text.B64ENC, and cc.util.Text.UUID_BUFFER.

Referenced by cc.ws.EventMgr.doSave(), and cc.ws.RopMgr.doSave().

Here is the caller graph for this function:



7.21.3.7 parseDouble() [1/2]

```
static double cc.util.Text.parseDouble (
            CharSequence iCharSeq )  [inline], [static]
```

Converts the character sequence into a double value.

Parameters

iCharSeq	a set of characters to be converted into a double value

Returns

the converted double value.

Definition at line 291 of file Text.java.

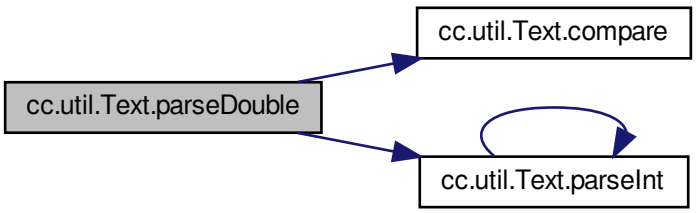
```
00291                                                       {
00292         // first test for expected string names
00293         if (compare(iCharSeq, POS_INF) == 0 ||
00294                 compare(iCharSeq, POS_INFINITY) == 0)
00295             return Double.POSITIVE_INFINITY;
00296
00297         if (compare(iCharSeq, NEG_INF) == 0 ||
00298                 compare(iCharSeq, NEG_INFINITY) == 0)
00299             return Double.NEGATIVE_INFINITY;
00300
00301         if (compare(iCharSeq, NAN) == 0)
00302             return Double.NaN;
00303
00304         int nExponent = 0;
00305         double dSign = 1.0;
```

```
00306         double dValue = 0.0;
00307         double dMultiplier = 0.1;
00308         double dFraction = 0.0;
00309 
00310         // valid characters for a double are -, +, ., digits, E, and e
00311         int nIndex = 0;
00312         int nState = INIT_CHAR;
00313         while (nIndex < iCharSeq.length() && nState != PARSE_END) {
00314             char cDigit = iCharSeq.charAt(nIndex++);
00315             switch (nState) {
00316                 // the digits test is first since it is the most likely
00317                 // parseInt cannot be used due to potential leading zeros
00318                 // in the decimal and fractional parts, i.e. -0.00763
00319                 case DECIMAL: {
00320                     if (Character.isDigit(cDigit)) {
00321                         // shift any existing value
00322                         dValue *= 10.0;
00323                         dValue += (cDigit - DIGIT_OFFSET);
00324                     } else {
00325                         // switch to other states for the double interpretation
00326                         switch (cDigit) {
00327                             case '.':
00328                                 nState = FRACTION;
00329                                 break;
00330 
00331                             case 'e':
00332                             case 'E':
00333                                 nState = EXPONENT;
00334                                 break;
00335 
00336                             default:
00337                                 nState = PARSE_END;
00338                                 break;
00339                         }
00340                     }
00341                 }
00342                 break;
00343 
00344                 case FRACTION: {
00345                     if (Character.isDigit(cDigit)) {
00346                         dFraction += (cDigit - DIGIT_OFFSET) * dMultiplier;
00347                         dMultiplier *= 0.1;
00348                     } else if (cDigit == 'e' || cDigit == 'E')
00349                         nState = EXPONENT;
00350                     else
00351                         nState = PARSE_END;
00352                 }
00353                 break;
00354 
00355                 // the exponent state is able to use the parseInt method
00356                 // as any fractional exponent will be ignored
00357                 case EXPONENT: {
00358                     nExponent = parseInt(iCharSeq, --nIndex, iCharSeq.length());
00359                     nState = PARSE_END;
00360                 }
00361                 break;
00362 
00363                 // the initial character test is only performed once
00364                 case INIT_CHAR: {
00365                     switch (cDigit) {
00366                         case '-': {
00367                             dSign = -1.0;
00368                             nState = DECIMAL;
00369                         }
00370                         break;
00371 
00372                         case '+': {
00373                             dSign = 1.0;
00374                             nState = DECIMAL;
00375                         }
00376                         break;
00377 
00378                         case '.':
00379                             nState = FRACTION;
00380                             break;
00381 
00382                         default: {
00383                             if (Character.isDigit(cDigit)) {
00384                                 // back up one character to test for digits
00385                                 --nIndex;
00386                                 nState = DECIMAL;
00387                             } else
00388                                 nState = PARSE_END;
00389                         }
00390                     }
00391                 }
00392             }
```

```
00393         }
00394
00395         // only generate an exponent when necessary
00396         double dExponent = 1.0;
00397         if (nExponent != 0) {
00398             // check the limits of the exponent
00399             if (nExponent < MIN_EXPONENT)
00400                 nExponent = MIN_EXPONENT;
00401
00402             if (nExponent > MAX_EXPONENT)
00403                 nExponent = MAX_EXPONENT;
00404
00405             dExponent = Math.pow(10.0, (double) nExponent);
00406         }
00407
00408         return (dSign * (dValue + dFraction) * dExponent);
00409     }
```

References cc.util.Text.compare(), cc.util.Text.DECIMAL, cc.util.Text.DIGIT_OFFSET, cc.util.Text.EXPONENT, cc.util.Text.FRACTION, cc.util.Text.INIT_CHAR, cc.util.Text.MAX_EXPONENT, cc.util.Text.MIN_EXPONENT, cc.util.Text.NAN, cc.util.Text.NEG_INF, cc.util.Text.NEG_INFINITY, cc.util.Text.PARSE_END, cc.util.Text.parseInt(), cc.util.Text.POS_INF, and cc.util.Text.POS_INFINITY.

Here is the call graph for this function:



7.21.3.8 parseDouble() [2/2]

```
static double cc.util.Text.parseDouble (
            CharSequence iCharSeq,
            int nPos,
            int nEndPos ) throws IndexOutOfBoundsException, NumberFormatException  [inline],
[static]
```

Definition at line 243 of file Text.java.

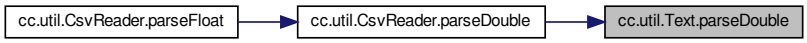
```
00245     {
00246         boolean bNeg = (iCharSeq.charAt(nPos) == '-');
00247         if (bNeg) // test for sign character
00248             ++nPos;
00249
00250         double dVal = 0.0;
00251         while (nPos < nEndPos && iCharSeq.charAt(nPos) != '.')
00252         {
00253             int nDigit = iCharSeq.charAt(nPos++) - DIGIT_OFFSET; // map char value
00254             if (nDigit < 0 || nDigit > 9) // check for non-numeric chars
00255                 throw new NumberFormatException();
00256
00257             dVal *= 10.0; // shift existing value
00258             dVal += nDigit; // add new digit value
00259         }
```

```
00260
00261         if (iCharSeq.charAt(nPos++) == '.') // fractional part check
00262         {
00263             double dDiv = 1.0;
00264             double dFrac = 0.0;
00265             while (nPos < nEndPos)
00266             {
00267                 int nDigit = iCharSeq.charAt(nPos++) - DIGIT_OFFSET; // map char value
00268                 if (nDigit < 0 || nDigit > 9) // check for non-numeric chars
00269                     throw new NumberFormatException();
00270
00271                 dDiv *= 10.0; // track decimal places
00272                 dFrac *= 10.0; // shift existing value
00273                 dFrac += nDigit; // add new digit value
00274             }
00275             dVal += dFrac / dDiv; // expensive division operator
00276         }
00277
00278         if (bNeg)
00279             return -dVal;
00280
00281         return dVal;
00282     }
```

References cc.util.Text.DIGIT_OFFSET.

Referenced by cc.util.CsvReader.parseDouble().

Here is the caller graph for this function:



7.21.3.9 parseInt() [1/2]

```
static int cc.util.Text.parseInt (
            CharSequence iCharSeq )  [inline], [static]
```

Wraps `Text#parseInt(java.lang.CharSequence, int, int)` to convert the sequence from beginning to end.

Parameters

iCharSeq	a set of characters to be converted into an integer value

Returns

the decimal integer value represented by the character sequence.

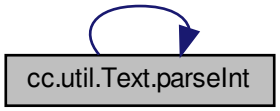
Definition at line 186 of file Text.java.

```
00186                                                   {
00187         return parseInt(iCharSeq, 0, iCharSeq.length());
00188     }
```

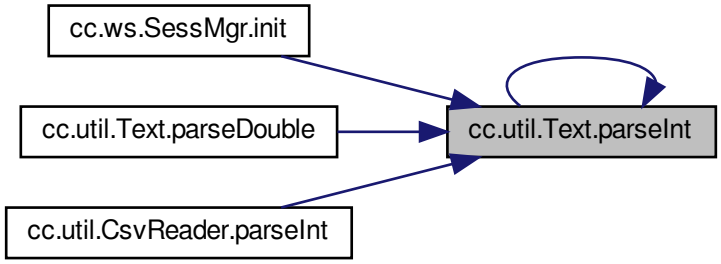
References cc.util.Text.parseInt().

Referenced by cc.ws.SessMgr.init(), cc.util.Text.parseDouble(), cc.util.Text.parseInt(), and cc.util.CsvReader.parseInt().

Here is the call graph for this function:



Here is the caller graph for this function:



7.21.3.10 parseInt() [2/2]

```
static int cc.util.Text.parseInt (
            CharSequence iCharSeq,
            int nPos,
            int nEndPos ) throws NumberFormatException  [inline], [static]
```

Parses the character sequence argument as a signed decimal integer. The characters in the sequence must all be decimal digits, except that the first character may be an ASCII minus sign '-' ('\u002D') to indicate a negative value.

Parameters

iCharSeq	a set of characters to be converted into an integer value
nPos	the position in the sequence where conversion begins
nEndPos	the sequence position where conversion stops (exclusive)

Returns

the decimal integer value represented by the character sequence

Exceptions

NumberFormatException	if the character sequence does not contain characters that can be converted to a decimal integer

Definition at line 204 of file Text.java.

```
00205                                           {
00206         int nSign = 1;
00207         int nValue = 0;
00208 
00209         // valid characters for an integer are -, +, and digits
00210         int nState = INIT_CHAR;
00211         while (nPos < nEndPos && nState != PARSE_END) {
00212             char cDigit = iCharSeq.charAt(nPos++);
00213             switch (nState) {
00214                 // the digits test is first since it is the most likely
00215                 case DECIMAL: {
00216                     if (Character.isDigit(cDigit)) {
00217                         // shift any existing value
00218                         nValue *= 10;
00219                         nValue += (cDigit - DIGIT_OFFSET);
00220                     } else
00221                         throw new NumberFormatException("Illegal character '" +
00222                                 iCharSeq.charAt(--nPos) + "' for integer " +
00223                                 "expression at position " + nPos + ".");
00224                 }
00225                 break;
00226 
00227                 // the initial character test is only performed once
00228                 case INIT_CHAR: {
00229                     nState = DECIMAL;
00230                     if (cDigit == '-')
00231                         nSign = -1;
00232                     else
00233                         // back up one character to test for digits
00234                         --nPos;
00235                 }
00236             }
00237         }
00238 
00239         return (nSign * nValue);
00240     }
```

References cc.util.Text.DECIMAL, cc.util.Text.DIGIT_OFFSET, cc.util.Text.INIT_CHAR, and cc.util.Text.PARSE_END.

7.21.3.11 parseLong() [1/2]

```
static long cc.util.Text.parseLong (
            CharSequence iCharSeq )  [inline], [static]
```

Wraps `Text#parseInt(java.lang.CharSequence, int, int)` to convert the sequence from beginning to end.

Parameters

iCharSeq	a set of characters to be converted into a long value

Returns

the decimal long value represented by the character sequence.

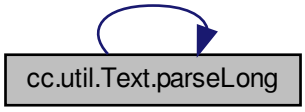
Definition at line 133 of file Text.java.

```
00133                                                   {
00134         return parseLong(iCharSeq, 0, iCharSeq.length());
00135     }
```

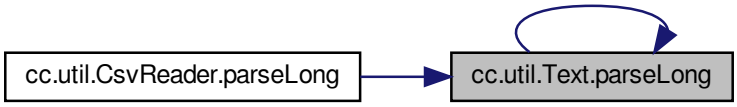
References cc.util.Text.parseLong().

Referenced by cc.util.Text.parseLong(), and cc.util.CsvReader.parseLong().

Here is the call graph for this function:



Here is the caller graph for this function:



7.21.3.12 parseLong() [2/2]

```
static long cc.util.Text.parseLong (
            CharSequence iCharSeq,
            int nPos,
            int nEndPos ) throws NumberFormatException  [inline], [static]
```

Parses the character sequence argument as a signed decimal long. The characters in the sequence must all be decimal digits, except that the first character may be an ASCII minus sign '-' ('\u002D') to indicate a negative value.

Parameters

iCharSeq	a set of characters to be converted into an integer value
nPos	the position in the sequence where conversion begins
nEndPos	the sequence position where conversion stops (exclusive)

Returns

the decimal long value represented by the character sequence

Exceptions

NumberFormatException	if the character sequence does not contain characters that can be converted to a decimal long

Definition at line 151 of file Text.java.

```
00152                                       {
00153         // test for the sign character
00154         boolean bNegative = (iCharSeq.charAt(nPos) == '-');
00155         if (bNegative)
00156             ++nPos;
00157 
00158         int nDigit = 0;
00159         long lValue = 0L;
00160         while (nPos < nEndPos) {
00161             // map the character value to the numeric value
00162             nDigit = iCharSeq.charAt(nPos++) - DIGIT_OFFSET;
00163             // test for characters that are not numbers
00164             if (nDigit < 0 || nDigit > 9)
00165                 throw new NumberFormatException();
00166 
00167             // shift the existing value and add the new digit value
00168             lValue *= 10L;
00169             lValue += nDigit;
00170         }
00171 
00172         if (bNegative)
00173             return -lValue;
00174 
00175         return lValue;
00176     }
```

References cc.util.Text.DIGIT_OFFSET.

7.21.3.13 removeWhitespace()

```
static void cc.util.Text.removeWhitespace (
             StringBuilder sBuffer )  [inline], [static]
```

Removes whitespace from the provided string builder.

Parameters

sBuffer	the string to remove whitespace from.

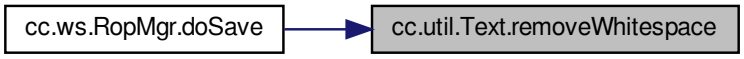
Definition at line 103 of file Text.java.

```
00103                                                           {
00104         // only remove whitespace if there is something to scan
00105         if (sBuffer.length() > 0) {
00106             // reverse iterate to the first non-whitespace character
00107             int nIndex = sBuffer.length();
00108             while (nIndex-- > 0 &&
00109                     Character.isWhitespace(sBuffer.charAt(nIndex))) ;
00110 
00111             // remove the trailing whitespace segment
00112             sBuffer.delete(++nIndex, sBuffer.length());
00113 
00114             // forward iterate to the first non-whitespace character
00115             nIndex = 0;
00116             while (nIndex < sBuffer.length() &&
00117                     Character.isWhitespace(sBuffer.charAt(nIndex++))) ;
```

```
00118
00119             // remove the leading whitespace segment
00120             if (nIndex > 0)
00121                 sBuffer.delete(0, --nIndex);
00122         }
00123     }
```

Referenced by cc.ws.RopMgr.doSave().

Here is the caller graph for this function:



7.21.3.14 replaceAll()

```
static void cc.util.Text.replaceAll (
             StringBuilder sBuffer,
             String sSearch,
             String sReplace )  [inline], [static]
```

Replaces all occurrences of the search string within the supplied buffer with the replacement string.

Parameters

sBuffer	StringBuilder buffer containing text to be searched.
sSearch	Search string to find in the buffer.
sReplace	Replacement string to substitute for the search string.

Definition at line 611 of file Text.java.

```
00612                                                                    {
00613         int nIndex = 0;
00614         while ((nIndex = sBuffer.indexOf(sSearch, nIndex)) >= 0) {
00615             sBuffer.replace(nIndex, nIndex + sSearch.length(), sReplace);
00616             // move start point after the replacement to enable recursion
00617             nIndex += sReplace.length();
00618         }
00619     }
```

7.21.3.15 startsWith()

```
static boolean cc.util.Text.startsWith (
             CharSequence iSource,
             CharSequence iPrefix )  [inline], [static]
```

Tests if the source character sequence begins with the prefix character sequence.

Parameters

iSource	the character sequence to check
iPrefix	the search character sequence

Returns

`true` if the initial source characters match the characters in the prefix. `false` otherwise.

Definition at line 477 of file Text.java.

```
00477                                                                                {
00478         int nIndex = iPrefix.length();
00479
00480         // the source cannot start with a pattern with more characters
00481         if (nIndex > iSource.length())
00482             return false;
00483
00484         // compare each characters starting at the end of the sequence
00485         boolean bMatch = true;
00486         while (bMatch && nIndex-- > 0)
00487             bMatch = (iSource.charAt(nIndex) == iPrefix.charAt(nIndex));
00488
00489         return bMatch;
00490     }
```

7.21.3.16 toHexString() [1/4]

```
static String cc.util.Text.toHexString (
            byte[] yBytes )  [inline], [static]
```

Converts a byte array into a hexadecimal string

Parameters

yBytes	Byte array containing data to be converted.

Returns

Hexadecimal string that represents the supplied byte data.

Definition at line 525 of file Text.java.

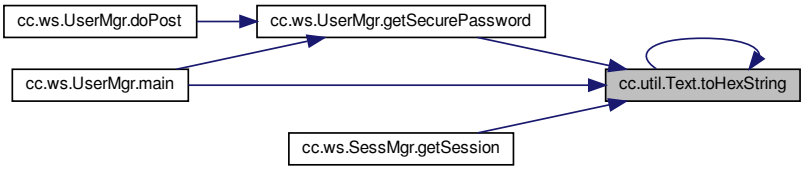
```
00526         {
00527             return toHexString(yBytes, 0, yBytes.length);
00528     }
```

References cc.util.Text.toHexString().

Referenced by cc.ws.UserMgr.getSecurePassword(), cc.ws.SessMgr.getSession(), cc.ws.UserMgr.main(), and cc.util.Text.toHexString().

Here is the call graph for this function:

Here is the caller graph for this function:



7.21.3.17 toHexString() [2/4]

```
static String cc.util.Text.toHexString (
            byte[] yBytes,
            int nOffset,
            int nLength )  [inline], [static]
```

Converts a byte array into a hexadecimal string

Parameters

yBytes	Byte array containing data to be converted.
nOffset	The position within the array to begin converting.
nLength	The number of bytes to be converted.

Returns

Hexadecimal string that represents the supplied byte data.

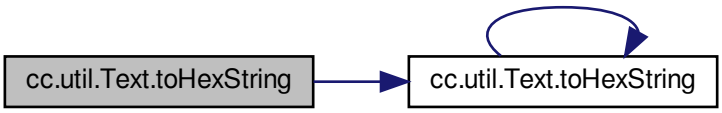
Definition at line 551 of file Text.java.

```
00552         {
00553             StringBuilder sBuf = new StringBuilder();
00554             toHexString(yBytes, nOffset, nLength, sBuf);
00555             return sBuf.toString();
```

```
00556         }
```

References cc.util.Text.toHexString().

Here is the call graph for this function:



7.21.3.18 toHexString() [3/4]

```
static void cc.util.Text.toHexString (
             byte[] yBytes,
             int nOffset,
             int nLength,
             StringBuilder sBuf )  [inline], [static]
```

Converts a byte array into a hexadecimal string

Parameters

yBytes	Byte array containing data to be converted.
nOffset	The position within the array to begin converting.
nLength	The number of bytes to be converted.
sBuf	String buffer that holds the converted characters.

Definition at line 567 of file Text.java.

```
00569         {
00570             for (; nOffset < nLength; nOffset++)
00571             {
00572                 sBuf.append(HEX_CHARS[((yBytes[nOffset] & 0xf0) » 4)]);
00573                 sBuf.append(HEX_CHARS[(yBytes[nOffset] & 0x0f)]);
00574             }
00575     }
```

References cc.util.Text.HEX_CHARS.

7.21.3.19 toHexString() [4/4]

```
static void cc.util.Text.toHexString (
             byte[] yBytes,
             StringBuilder sBuf )  [inline], [static]
```

Converts a byte array into a hexadecimal string

Parameters

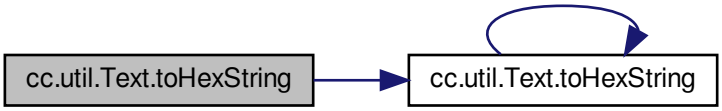
yBytes	Byte array containing data to be converted.
sBuf	String buffer that holds the converted characters.

Definition at line 537 of file Text.java.

```
00538         {
00539             toHexString(yBytes, 0, yBytes.length, sBuf);
00540     }
```

References cc.util.Text.toHexString().

Here is the call graph for this function:



7.21.3.20 truncate()

```
static String cc.util.Text.truncate (
             String sValue,
             int nLength )  [inline], [static]
```

Truncates the passed string value if it is longer than the passed length, or returns the original value if it isn't.

Parameters

sValue	The value to truncate
nLength	The maximum length of the truncated string.

Returns

The truncated string

Definition at line 652 of file Text.java.

```
00652                                                                   {
00653         if(sValue == null || sValue.length() <= nLength)
00654           return sValue;
00655         else
00656           return sValue.substring(0, nLength);
00657     }
```

7.21.4 Member Data Documentation

7.21.4.1 B64ENC

```
final Base64.Encoder cc.util.Text.B64ENC = Base64.getUrlEncoder().withoutPadding()  [static]
```

Convenience Base64 URL encoder without padding

Definition at line 82 of file Text.java.

Referenced by cc.util.Text.getUUID().

7.21.4.2 DECIMAL

```
final int cc.util.Text.DECIMAL = 1  [static], [private]
```

State Constant marking decimal.

Definition at line 42 of file Text.java.

Referenced by cc.util.Text.parseDouble(), and cc.util.Text.parseInt().

7.21.4.3 DIGIT_OFFSET

```
final int cc.util.Text.DIGIT_OFFSET = 48  [static], [private]
```

Digit offset.

Definition at line 25 of file Text.java.

Referenced by cc.util.Text.parseDouble(), cc.util.Text.parseInt(), and cc.util.Text.parseLong().

7.21.4.4 EXPONENT

```
final int cc.util.Text.EXPONENT = 3  [static], [private]
```

State Constant marking exponents.

Definition at line 50 of file Text.java.

Referenced by cc.util.Text.parseDouble().

7.21.4.5 FRACTION

```
final int cc.util.Text.FRACTION = 2  [static], [private]
```

State Constant marking fraction.

Definition at line 46 of file Text.java.

Referenced by cc.util.Text.parseDouble().

7.21.4.6 HEX_CHARS

```
final char [] cc.util.Text.HEX_CHARS  [static], [private]
```

Initial value:

```
=
            {
                    '0', '1', '2', '3', '4', '5', '6', '7',
                    '8', '9', 'a', 'b', 'c', 'd', 'e', 'f'
            }
```

Lower-case character set used to encode a byte array as a hex string.

Definition at line 17 of file Text.java.

Referenced by cc.util.Text.toHexString().

7.21.4.7 INIT_CHAR

```
final int cc.util.Text.INIT_CHAR = 0  [static], [private]
```

State Constant marking initial character.

Definition at line 38 of file Text.java.

Referenced by cc.util.Text.parseDouble(), and cc.util.Text.parseInt().

7.21.4.8 MAX_EXPONENT

```
final int cc.util.Text.MAX_EXPONENT = 308  [static], [private]
```

Maximum exponent limit.

Definition at line 33 of file Text.java.

Referenced by cc.util.Text.parseDouble().

7.21.4.9 MIN_EXPONENT

```
final int cc.util.Text.MIN_EXPONENT = -323  [static], [private]
```

Minimum exponent limit.

Definition at line 29 of file Text.java.

Referenced by cc.util.Text.parseDouble().

7.21.4.10 NAN

```
final String cc.util.Text.NAN = "NaN"  [static], [private]
```

Not a number - string name constant used to convert strings to doubles

Definition at line 78 of file Text.java.

Referenced by cc.util.Text.parseDouble().

7.21.4.11 NEG_INF

```
final String cc.util.Text.NEG_INF = "-\u221E"  [static], [private]
```

Negative infinity - unicode constant used to convert strings to doubles

Definition at line 63 of file Text.java.

Referenced by cc.util.Text.parseDouble().

7.21.4.12 NEG_INFINITY

```
final String cc.util.Text.NEG_INFINITY = "-Infinity"  [static], [private]
```

Negative infinity - string name constant used to convert strings to doubles

Definition at line 73 of file Text.java.

Referenced by cc.util.Text.parseDouble().

7.21.4.13 PARSE_END

```
final int cc.util.Text.PARSE_END = 4  [static], [private]
```

State Constant marking end of string.

Definition at line 54 of file Text.java.

Referenced by cc.util.Text.parseDouble(), and cc.util.Text.parseInt().

7.21.4.14 POS_INF

```
final String cc.util.Text.POS_INF = "\u221E"  [static], [private]
```

Positive infinity - unicode constant used to convert strings to doubles

Definition at line 59 of file Text.java.

Referenced by cc.util.Text.parseDouble().

7.21.4.15 POS_INFINITY

```
final String cc.util.Text.POS_INFINITY = "Infinity"  [static], [private]
```

Positive infinity - string name constant used to convert strings to doubles

Definition at line 68 of file Text.java.

Referenced by cc.util.Text.parseDouble().

7.21.4.16 UUID_BUFFER

```
final ByteBuffer cc.util.Text.UUID_BUFFER = ByteBuffer.allocate(16)  [static], [private]
```

Definition at line 84 of file Text.java.

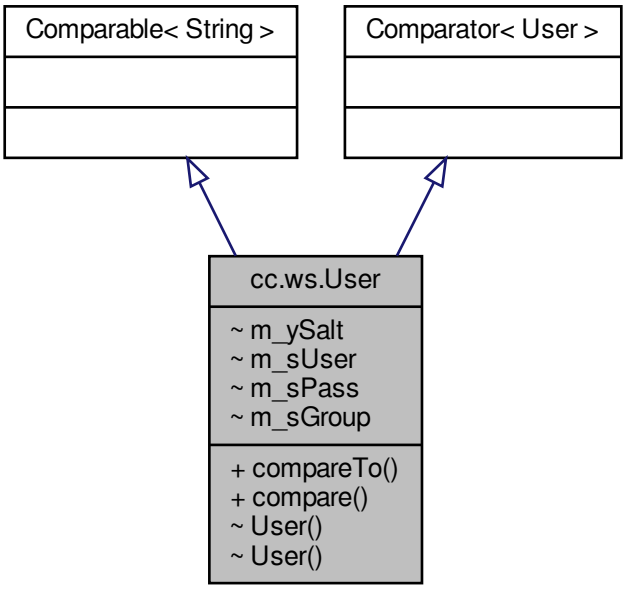
Referenced by cc.util.Text.getUUID().

The documentation for this class was generated from the following file:

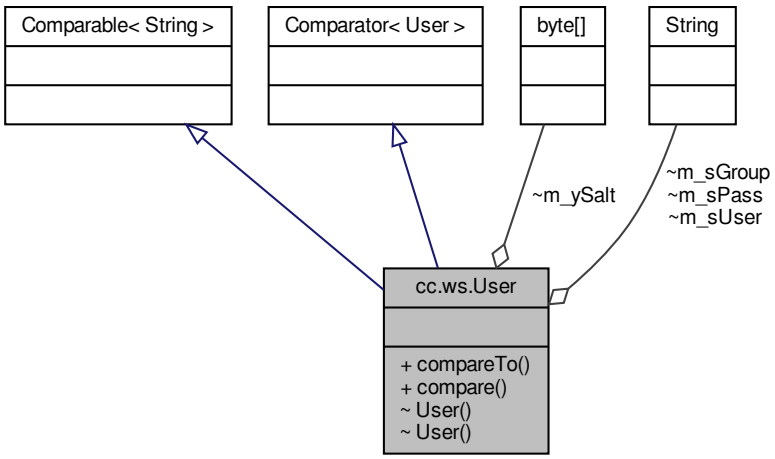
- src/cc/util/Text.java

7.22 cc.ws.User Class Reference

Inheritance diagram for cc.ws.User:



Collaboration diagram for cc.ws.User:



Public Member Functions

- int compareTo (String sUser)
- int compare (User oLhs, User oRhs)

Package Functions

- User ()
- User (CsvReader oCsv)

Package Attributes

- byte[ ] m_ySalt
- String m_sUser
- String m_sPass
- String m_sGroup

7.22.1 Detailed Description

Definition at line 8 of file User.java.

7.22.2 Constructor & Destructor Documentation

7.22.2.1 User() [1/2]

```
cc.ws.User.User ( )  [inline], [package]
```

Definition at line 16 of file User.java.

```
00017     {
00018     }
```

7.22.2.2 User() [2/2]

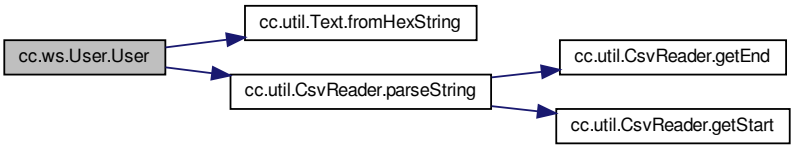
```
cc.ws.User.User (
            CsvReader oCsv )  [inline], [package]
```

Definition at line 21 of file User.java.

```
00022     {
00023         StringBuilder sCol = new StringBuilder();
00024         oCsv.parseString(sCol, 0);
00025         m_sUser = sCol.toString(); // save username
00026 
00027         oCsv.parseString(sCol, 1);
00028         m_ySalt = Text.fromHexString(sCol);
00029 
00030         oCsv.parseString(sCol, 2);
00031         m_sPass = sCol.toString(); // keep password as hexadecimal string
00032 
00033         oCsv.parseString(sCol, 3);
00034         m_sGroup = sCol.toString().intern(); // very few group patterns
00035     }
```

References cc.util.Text.fromHexString(), cc.ws.User.m_sGroup, cc.ws.User.m_sPass, cc.ws.User.m_sUser, cc.ws.User.m_ySalt, and cc.util.CsvReader.parseString().

Here is the call graph for this function:



7.22.3 Member Function Documentation

7.22.3.1 compare()

```
int cc.ws.User.compare (
            User oLhs,
            User oRhs )  [inline]
```

Definition at line 46 of file User.java.

```
00047     {
00048         return oLhs.m_sUser.compareTo(oRhs.m_sUser);
00049     }
```

References cc.ws.User.m_sUser.

7.22.3.2 compareTo()

```
int cc.ws.User.compareTo (
            String sUser )  [inline]
```

Definition at line 39 of file User.java.

```
00040     {
00041         return m_sUser.compareTo(sUser);
00042     }
```

References cc.ws.User.m_sUser.

7.22.4 Member Data Documentation

7.22.4.1 m_sGroup

```
String cc.ws.User.m_sGroup  [package]
```

Definition at line 13 of file User.java.

Referenced by cc.ws.User.User().

7.22.4.2 m_sPass

```
String cc.ws.User.m_sPass  [package]
```

Definition at line 12 of file User.java.

Referenced by cc.ws.User.User(), and cc.ws.UserMgr.doPost().

7.22.4.3 m_sUser

```
String cc.ws.User.m_sUser  [package]
```

Definition at line 11 of file User.java.

Referenced by cc.ws.User.User(), cc.ws.User.compare(), and cc.ws.User.compareTo().

7.22.4.4 m_ySalt

```
byte [] cc.ws.User.m_ySalt  [package]
```

Definition at line 10 of file User.java.

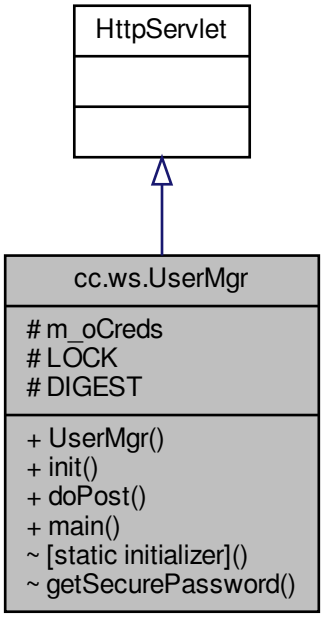
Referenced by cc.ws.User.User(), and cc.ws.UserMgr.doPost().

The documentation for this class was generated from the following file:

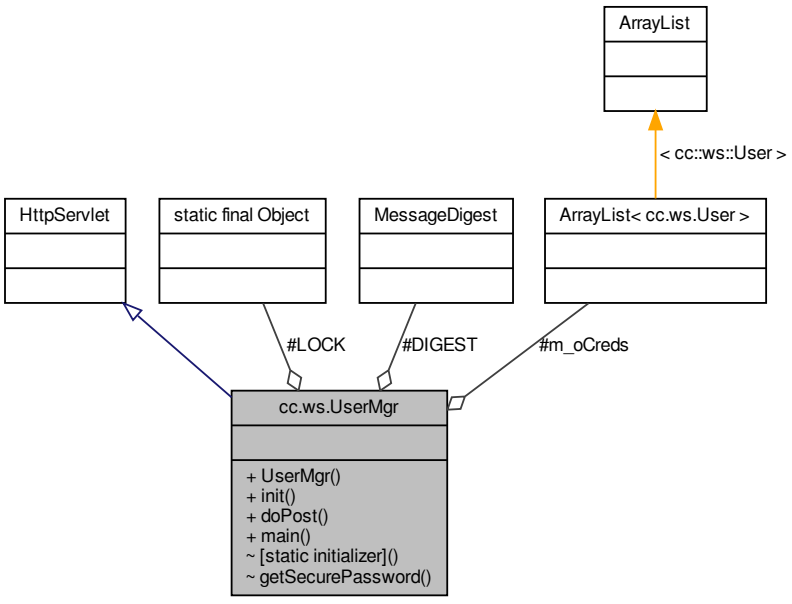
- src/cc/ws/User.java

7.23 cc.ws.UserMgr Class Reference

Inheritance diagram for cc.ws.UserMgr:



Collaboration diagram for cc.ws.UserMgr:



Public Member Functions

- UserMgr ()
- void init ()
- void doPost (HttpServletRequest oReq, HttpServletResponse oRep)

Static Public Member Functions

- static void main (String[ ] sArgs)

Protected Attributes

- ArrayList< User > m_oCreds = new ArrayList()

Static Protected Attributes

- static final Object LOCK = new Object()
- static MessageDigest DIGEST

Static Package Functions

- [static initializer]
- static void getSecurePassword (String sPass, byte[ ] ySalt, StringBuilder sBuf)

7.23.1 Detailed Description

Definition at line 16 of file UserMgr.java.

7.23.2 Constructor & Destructor Documentation

7.23.2.1 UserMgr()

```
cc.ws.UserMgr.UserMgr ( )  [inline]
```

Definition at line 33 of file UserMgr.java.

```
00034     {
00035     }
```

7.23.3 Member Function Documentation

7.23.3.1 [static initializer]()

```
cc.ws.UserMgr.[static initializer]  [inline], [static], [package]
```

7.23.3.2 doPost()

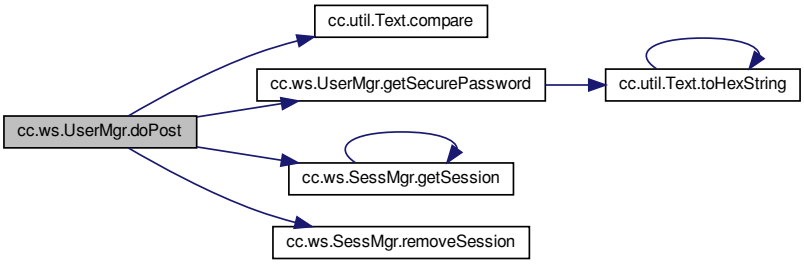
```
void cc.ws.UserMgr.doPost (
            HttpServletRequest oReq,
            HttpServletResponse oRep )  [inline]
```

Definition at line 55 of file UserMgr.java.

```
00056     {
00057         StringBuilder sBuf = new StringBuilder("{\n");
00058         String sPath = oReq.getPathInfo();
00059         if (sPath.contains("login"))
00060         {
00061             String sUname = oReq.getParameter("uname");
00062             String sPword = oReq.getParameter("pword");
00063             if (sUname != null && sUname.length() > 0 && sPword != null && sPword.length() > 0)
00064             {
00065                 int nIndex = Collections.binarySearch(m_oCreds, sUname);
00066                 if (nIndex >= 0)
00067                 {
00068                     User oUser = m_oCreds.get(nIndex);
00069                     StringBuilder sSecPass = new StringBuilder();
00070                     getSecurePassword(sPword, oUser.m_ySalt, sSecPass);
00071                     if (Text.compare(oUser.m_sPass, sSecPass) == 0)
00072                     {
00073                         Session oSess = SessMgr.getSession(oReq, true);
00074                         oSess.m_oUser = oUser; // save credentials in session
00075                         sBuf.append("\t\"token\": \"").append(oSess.m_sToken).append("\"\n");
00076                     }
00077                 }
00078             }
00079         }
00080         else
00081         {
00082             Session oSess = SessMgr.getSession(oReq);
00083             if (sPath.contains("check"))
00084             {
00085                 if (oSess != null)
00086                     sBuf.append("\t\"token\": \"").append(oSess.m_sToken).append("\"\n");
00087             }
00088             else if (sPath.contains("logout"))
00089             {
00090                     SessMgr.removeSession(oSess);
00091             }
00092             else if (sPath.contains("update"))
00093             {
00094             }
00095         }
00096         sBuf.append("}\n");
00097
00098         try (ServletOutputStream oOut = oRep.getOutputStream())
00099         {
00100             for (int nIndex = 0; nIndex < sBuf.length(); nIndex++)
00101                 oOut.print(sBuf.charAt(nIndex));
00102         }
00103         catch (Exception oEx)
00104         {
00105         }
00106     }
```

References cc.util.Text.compare(), cc.ws.UserMgr.getSecurePassword(), cc.ws.SessMgr.getSession(), cc.ws.UserMgr.m_oCreds, cc.ws.User.m_sPass, cc.ws.Session.m_sToken, cc.ws.User.m_ySalt, and cc.ws.SessMgr.removeSession().

Here is the call graph for this function:



7.23.3.3 getSecurePassword()

```
static void cc.ws.UserMgr.getSecurePassword (
            String sPass,
            byte[] ySalt,
            StringBuilder sBuf )  [inline], [static], [package]
```

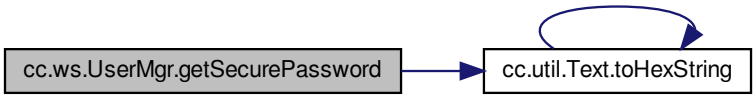
Definition at line 109 of file UserMgr.java.

```
00110     {
00111         synchronized(LOCK)
00112         {
00113             DIGEST.reset();
00114             DIGEST.update(ySalt);
00115             Text.toHexString(DIGEST.digest(sPass.getBytes()), sBuf);
00116         }
00117     }
```

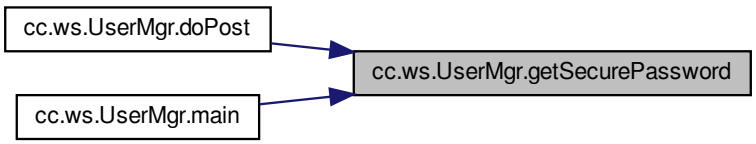
References cc.ws.UserMgr.DIGEST, cc.ws.UserMgr.LOCK, and cc.util.Text.toHexString().

Referenced by cc.ws.UserMgr.doPost(), and cc.ws.UserMgr.main().

Here is the call graph for this function:



Here is the caller graph for this function:



7.23.3.4 init()

```
void cc.ws.UserMgr.init ( )  [inline]
```

Definition at line 39 of file UserMgr.java.

```
00040     {
00041         ServletConfig oConf = getServletConfig();
00042         try (CsvReader oCsv = new CsvReader(new FileInputStream(oConf.getInitParameter("pwdfile"))))
00043         {
00044             while (oCsv.readLine() > 0)
00045                 m_oCreds.add(new User(oCsv));
00046         }
00047         catch (Exception oEx)
00048         {
00049         }
00050         Collections.sort(m_oCreds, new User());
00051     }
```

References cc.ws.UserMgr.m_oCreds.

7.23.3.5 main()

```
static void cc.ws.UserMgr.main (
             String[] sArgs )  [inline], [static]
```

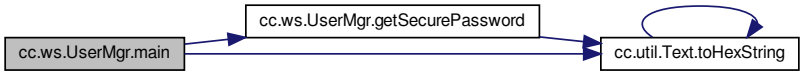
Definition at line 120 of file UserMgr.java.

```
00121     {
00122         String sUser = sArgs[0];
00123         String sPass = sArgs[1];
00124 
00125         byte[] ySalt = new byte[32]; // use 256-bit algorithm
00126         try
00127         {
00128             java.security.SecureRandom.getInstance("SHA1PRNG").nextBytes(ySalt);
00129             StringBuilder sBuf = new StringBuilder();
00130             UserMgr.getSecurePassword(sPass, ySalt, sBuf);
00131 
00132             System.out.print(sUser);
00133             System.out.print(",");
00134             System.out.print(Text.toHexString(ySalt));
00135             System.out.print(",");
00136             System.out.print(sBuf.toString());
00137             System.out.print(",");
00138             System.out.println("abcdefghijklmnopqrstWvwxyz");
00139         }
00140         catch (Exception oEx)
00141         {
```

```
00142         }
00143     }
```

References cc.ws.UserMgr.getSecurePassword(), and cc.util.Text.toHexString().

Here is the call graph for this function:



7.23.4 Member Data Documentation

7.23.4.1 DIGEST

```
MessageDigest cc.ws.UserMgr.DIGEST  [static], [protected]
```

Definition at line 19 of file UserMgr.java.

Referenced by cc.ws.UserMgr.getSecurePassword().

7.23.4.2 LOCK

```
final Object cc.ws.UserMgr.LOCK = new Object()  [static], [protected]
```

Definition at line 18 of file UserMgr.java.

Referenced by cc.ws.UserMgr.getSecurePassword().

7.23.4.3 m_oCreds

```
ArrayList<User> cc.ws.UserMgr.m_oCreds = new ArrayList()  [protected]
```

Definition at line 20 of file UserMgr.java.

Referenced by cc.ws.UserMgr.doPost(), and cc.ws.UserMgr.init().

The documentation for this class was generated from the following file:

- src/cc/ws/UserMgr.java

Chapter 8

File Documentation

8.1 README.md File Reference

8.2 src/cc/geosrv/Mercator.java File Reference

Classes

- class cc.geosrv.Mercator

Packages

- package cc.geosrv

8.3 Mercator.java

Go to the documentation of this file.

```
00001 package cc.geosrv;
00002
00003 /*
00004  * To change this license header, choose License Headers in Project Properties.
00005  * To change this template file, choose Tools | Templates
00006  * and open the template in the editor.
00007  */
00008
00013 public class Mercator
00014 {
00015     private static final int[] POW = new int[24];
00016     private final double[] RES = new double[24];
00017     private static final double R_MAJOR = 6378137.0;
00018     private static final double R_MINOR = 6356752.3142;
00019     private static final double R_RATIO = R_MINOR / R_MAJOR;
00020     private static final double ECC = Math.sqrt(1.0 - (R_RATIO * R_RATIO));
00021     private static final double ECC_OVER_TWO = ECC / 2.0;
00022     public static final double PI_OVER_TWO = Math.PI / 2.0;
00023     private static final double ORIGIN_SHIFT = Math.PI * R_MAJOR;
00024     private static final double ORIGIN_SHIFT_DIVIDED_BY_180 = ORIGIN_SHIFT / 180.0;
00025     private static final double PI_OVER_180 = Math.PI / 180.0;
00026     private static final double PI_OVER_360 = PI_OVER_180 / 2.0;
00027     public static final double MAX_LAT = 85.05112877980659;
00028     public static final double MIN_LAT = -MAX_LAT;
00029     public static final double MAX_LON = 180;
00030     public static final double MIN_LON = -MAX_LON;
00031
00032     int m_nTileSize;
```

```
00033     double m_dInitRes;
00034
00035
00036     static
00037     {
00038         for (int nIndex = 0; nIndex < POW.length; nIndex++)
00039             POW[nIndex] = (int)Math.pow(2.0, nIndex);
00040     }
00041
00042
00043     public Mercator()
00044     {
00045         this(256);
00046     }
00047
00048
00049     public Mercator(int nTileSize)
00050     {
00051         m_nTileSize = nTileSize;
00052         m_dInitRes = 2.0 * ORIGIN_SHIFT / m_nTileSize;
00053         for (int nIndex = 0; nIndex < RES.length; nIndex++)
00054             RES[nIndex] = m_dInitRes / POW[nIndex];
00055     }
00056
00057
00058     public static int getExtent(int nZoom)
00059     {
00060         return POW[nZoom] * 256;
00061     }
00062
00063
00064     public static double lonToMeters(double dLon)
00065     {
00066         return dLon * ORIGIN_SHIFT_DIVIDED_BY_180;
00067     }
00068
00069
00070     public static double latToMeters(double dLat)
00071     {
00072         return Math.log(Math.tan((90.0 + dLat) * PI_OVER_360)) * R_MAJOR;
00073     }
00074
00075
00076     public static double xToLon(double dX)
00077     {
00078         return dX / ORIGIN_SHIFT * 180.0;
00079     }
00080
00081
00082     public static double yToLat(double dY)
00083     {
00084         double dLat = dY / ORIGIN_SHIFT * 180.0;
00085         return 180.0 / Math.PI * (2 * Math.atan(Math.exp(dLat * PI_OVER_180)) - PI_OVER_TWO);
00086     }
00087
00088
00089     public static void lonLatToMeters(double dLon, double dLat, double[] dMeters)
00090     {
00091         dMeters[0] = lonToMeters(dLon);
00092         dMeters[1] = latToMeters(dLat);
00093     }
00094
00095
00096     public void metersToLonLat(double dX, double dY, double[] dLatLon)
00097     {
00098         dLatLon[0] = dX / ORIGIN_SHIFT * 180.0;
00099         dLatLon[1] = dY / ORIGIN_SHIFT * 180.0;
00100         dLatLon[1] = 180.0 / Math.PI * (2 * Math.atan(Math.exp(dLatLon[1] * PI_OVER_180)) -
      PI_OVER_TWO);
00101     }
00102
00103     public void pixelsToMeters(double dXp, double dYp, int nZoom, double[] dMeters)
00104     {
00105         double dRes = resolution(nZoom);
00106         dMeters[0] = dXp * dRes - ORIGIN_SHIFT;
00107         dMeters[1] = -(dYp * dRes - ORIGIN_SHIFT);
00108     }
00109
00110     public void metersToPixels(double dXm, double dYm, int nZoom, double[] dPixels)
00111     {
00112         double dRes = resolution(nZoom);
00113         dPixels[0] = (dXm + ORIGIN_SHIFT) / dRes;
00114         dPixels[1] = (dYm + ORIGIN_SHIFT) / dRes;
00115     }
00116
00117     public void pixelsToTile(double dXp, double dYp, int[] nTiles)
00118     {
```

```
00119         nTiles[0] = (int)((Math.ceil(dXp / m_nTileSize)) - 1);
00120         nTiles[1] = (int)((Math.ceil(dYp / m_nTileSize)) - 1);
00121     }
00122
00123     public void tileBounds(double dXt, double dYt, int nZoom, double[] dBounds)
00124     {
00125         double[] dMeters = new double[2];
00126         pixelsToMeters(dXt * m_nTileSize, dYt * m_nTileSize, nZoom, dMeters);
00127         dBounds[0] = dMeters[0];
00128         dBounds[3] = dMeters[1];
00129         pixelsToMeters((dXt + 1) * m_nTileSize, (dYt + 1) * m_nTileSize, nZoom, dMeters);
00130         dBounds[2] = dMeters[0];
00131         dBounds[1] = dMeters[1];
00132     }
00133
00134     public void lonLatBounds(double dXt, double dYt, int nZoom, double[] dBounds)
00135     {
00136         double[] dMeterBounds = new double[4];
00137         tileBounds(dXt, dYt, nZoom, dMeterBounds);
00138         double[] dLonLat = new double[2];
00139         metersToLonLat(dMeterBounds[0], dMeterBounds[1], dLonLat);
00140         dBounds[0] = dLonLat[0];
00141         dBounds[1] = dLonLat[1];
00142         metersToLonLat(dMeterBounds[2], dMeterBounds[3], dLonLat);
00143         dBounds[2] = dLonLat[0];
00144         dBounds[3] = dLonLat[1];
00145     }
00146
00147
00148     public void lonLatToTile(double dLon, double dLat, int nZoom, int[] nTiles)
00149     {
00150         double[] dTemp = new double[2];
00151         lonLatToMeters(dLon, dLat, dTemp);
00152         metersToPixels(dTemp[0], dTemp[1], nZoom, dTemp);
00153         pixelsToTile(dTemp[0], dTemp[1], nTiles);
00154         nTiles[1] = POW[nZoom] - nTiles[1] - 1;
00155     }
00156
00157
00158     public void metersToTile(double dXm, double dYm, int nZoom, int[] nTiles)
00159     {
00160         double[] dPixels = new double[2];
00161         metersToPixels(dXm, dYm, nZoom, dPixels);
00162         pixelsToTile(dPixels[0], dPixels[1], nTiles);
00163         nTiles[1] = POW[nZoom] - nTiles[1] - 1;
00164     }
00165
00166
00167     public double resolution(int nZoom)
00168     {
00169 //      return m_dInitRes / POW[nZoom];
00170         return RES[nZoom];
00171     }
00172
00173
00174     public static double eMercX(double lon) {
00175         return R_MAJOR * Math.toRadians(lon);
00176     }
00177
00178     public static double eMercY(double lat) {
00179         if (lat > 89.5) {
00180             lat = 89.5;
00181         }
00182         if (lat < -89.5) {
00183             lat = -89.5;
00184         }
00185         double phi = Math.toRadians(lat);
00186         double con = ECC * Math.sin(phi);
00187         con = Math.pow(((1.0-con)/(1.0+con)), ECC_OVER_TWO);
00188         double ts = Math.tan(0.5 * ((PI_OVER_TWO) - phi))/con;
00189         double y = 0 - R_MAJOR * Math.log(ts);
00190         return y;
00191     }
00192
00193
00194     public static double eLon(double dX)
00195     {
00196         return Math.toDegrees(dX / R_MAJOR);
00197     }
00198
00199
00200     public static double eLat(double dY)
00201     {
00202         double ts = Math.exp(-dY / R_MAJOR);
00203         double phi = PI_OVER_TWO - 2 * Math.atan(ts);
00204         double dphi = 1.0;
00205         for (int i = 0; Math.abs(dphi) > 0.000000001 && i < 15; i++)
```

```
00206         {
00207             double con = ECC * Math.sin(phi);
00208             dphi = PI_OVER_TWO - 2 * Math.atan(ts * Math.pow((1.0 - con) / (1.0 + con), ECC_OVER_TWO))
      - phi;
00209             phi += dphi;
00210         }
00211
00212         return Math.toDegrees(phi);
00213     }
00214 }
```

8.4 src/cc/util/Arrays.java File Reference

Classes

- class cc.util.Arrays
- class cc.util.Arrays.GroupIterator
- class cc.util.Arrays.DoubleGroupIterator
- class cc.util.Arrays.IntGroupIterator

Packages

- package cc.util

8.5 Arrays.java

Go to the documentation of this file.

```
00001 package cc.util;
00002
00003 import java.io.PrintStream;
00004 import java.util.Iterator;
00005
00006
00011 public abstract class Arrays
00012 {
00013     private static final int DEFAULT_CAPACITY = 12;
00014
00015
00016     private Arrays()
00017     {
00018     }
00019
00020
00021     public static double[] newDoubleArray()
00022     {
00023         return newDoubleArray(DEFAULT_CAPACITY);
00024     }
00025
00026
00027     public static double[] newDoubleArray(int nCapacity)
00028     {
00029         double[] dVals = new double[++nCapacity]; // reserve slot for size
00030         dVals[0] = 1.0; // initial position is always one
00031         return dVals;
00032     }
00033
00034
00035     public static Iterator<double[]> iterator(double[] dSrc, double[] dDest, int nStart, int nStep)
00036     {
00037         return new DoubleGroupIterator(dSrc, dDest, nStart, nStep);
00038     }
00039
00040
00041     public static double[] ensureCapacity(double[] dVals, int nDemand)
00042     {
00043         if ((int)dVals[0] + nDemand < dVals.length)
00044             return dVals; // no changes needed
00045
```

```
00046         double[] dNew = new double[nDemand + (3 * dVals.length » 1)];
00047         System.arraycopy(dVals, 0, dNew, 0, (int)dVals[0]);
00048         return dNew;
00049     }
00050 
00051 
00052     public static int size(double[] dVals)
00053     {
00054         return (int)dVals[0];
00055     }
00056 
00057 
00058     public static double[] add(double[] dVals, double d1)
00059     {
00060         dVals = ensureCapacity(dVals, 1);
00061         dVals[(int)dVals[0]] = d1; // current insertion position
00062         dVals[0] += 1.0; // update position
00063         return dVals;
00064     }
00065 
00066 
00067     public static double[] add(double[] dVals, double d1, double d2)
00068     {
00069         dVals = ensureCapacity(dVals, 2);
00070         int nIndex = (int)dVals[0]; // current insertion position
00071         dVals[nIndex++] = d1;
00072         dVals[nIndex++] = d2;
00073         dVals[0] = (double)nIndex; // track insertion position
00074         return dVals;
00075     }
00076 
00077 
00078     public static double[] add(double[] dVals, double[] dMore)
00079     {
00080         dVals = ensureCapacity(dVals, dMore.length);
00081         int nIndex = (int)dVals[0]; // current insertion position
00082         System.arraycopy(dMore, 0, dVals, nIndex, dMore.length);
00083         dVals[0] = nIndex + dMore.length; // track insertion position
00084         return dVals;
00085     }
00086 
00087 
00088     public static int[] newIntArray()
00089     {
00090         return newIntArray(DEFAULT_CAPACITY);
00091     }
00092 
00093 
00094     public static int[] newIntArray(int nCapacity)
00095     {
00096         int[] nVals = new int[++nCapacity]; // reserve slot for size
00097         nVals[0] = 1; // initial position is always one
00098         return nVals;
00099     }
00100 
00101 
00102     public static Iterator<int[]> iterator(int[] nSrc, int[] nDest, int nStart, int nStep)
00103     {
00104         return new IntGroupIterator(nSrc, nDest, nStart, nStep);
00105     }
00106 
00107 
00108     public static int[] ensureCapacity(int[] nVals, int nDemand)
00109     {
00110         if (nVals[0] + nDemand < nVals.length)
00111             return nVals; // no changes needed
00112 
00113         int[] nNew = new int[nDemand + (3 * nVals.length » 1)];
00114         System.arraycopy(nVals, 0, nNew, 0, nVals[0]);
00115         return nNew;
00116     }
00117 
00118 
00119     public static int size(int[] nVals)
00120     {
00121         return nVals[0];
00122     }
00123 
00124 
00125     public static int[] add(int[] nVals, int n1)
00126     {
00127         nVals = ensureCapacity(nVals, 1);
00128         nVals[nVals[0]] = n1; // current insertion position
00129         ++nVals[0]; // update position
00130         return nVals;
00131     }
00132 
```

```
00133
00134     public static int[] add(int[] nVals, int n1, int n2)
00135     {
00136         nVals = ensureCapacity(nVals, 2);
00137         int nIndex = nVals[0]; // current insertion position
00138         nVals[nIndex++] = n1;
00139         nVals[nIndex++] = n2;
00140         nVals[0] = nIndex; // track insertion position
00141         return nVals;
00142     }
00143
00144
00145     public static int[] add(int[] nVals, int[] nMore)
00146     {
00147         nVals = ensureCapacity(nVals, nMore.length);
00148         int nIndex = nVals[0]; // current insertion position
00149         System.arraycopy(nMore, 0, nVals, nIndex, nMore.length);
00150         nVals[0] = nIndex + nMore.length; // track insertion position
00151         return nVals;
00152     }
00153
00154
00155     public static void printArray(double[] dArray, int nStart, PrintStream oPrint) throws Exception
00156     {
00157         Iterator<double[]> oIt = iterator(dArray, new double[1], nStart, 1);
00158         boolean bWrite = oIt.hasNext();
00159         if (bWrite)
00160         {
00161             double[] dVal = oIt.next();
00162             oPrint.append(Double.toString(dVal[0]));
00163         }
00164         while (oIt.hasNext())
00165         {
00166             double[] dVal = oIt.next();
00167             oPrint.append(",").append(Double.toString(dVal[0]));
00168         }
00169         if (bWrite)
00170             oPrint.append("\n");
00171     }
00172
00173
00174     public static void printArray(int[] nArray, int nStart, PrintStream oPrint) throws Exception
00175     {
00176         Iterator<int[]> oIt = iterator(nArray, new int[1], nStart, 1);
00177         boolean bWrite = oIt.hasNext();
00178         if (bWrite)
00179         {
00180             int[] nVal = oIt.next();
00181             oPrint.append(Integer.toString(nVal[0]));
00182         }
00183         while (oIt.hasNext())
00184         {
00185             int[] nVal = oIt.next();
00186             oPrint.append(",").append(Integer.toString(nVal[0]));
00187         }
00188         if (bWrite)
00189             oPrint.append("\n");
00190     }
00191
00192
00193     private static abstract class GroupIterator
00194     {
00195         protected int m_nPos;
00196         protected int m_nEnd;
00197         protected int m_nStep;
00198
00199
00200         protected GroupIterator()
00201         {
00202         }
00203
00204
00205         protected GroupIterator(int nStart, int nLimit, int nDestSize, int nStep)
00206         {
00207             m_nStep = nStep;
00208             m_nEnd = nLimit - nDestSize; // array end boundary
00209             m_nPos = nStart;
00210         }
00211
00212
00213         public boolean hasNext()
00214         {
00215             return (m_nPos <= m_nEnd);
00216         }
00217
00218
00219         public void remove()
```

```
00220         {
00221             throw new UnsupportedOperationException("remove");
00222         }
00223     }
00224
00225
00226     private static class DoubleGroupIterator extends GroupIterator implements Iterator<double[]>
00227     {
00228         private double[] m_dSrc;
00229         private double[] m_dDest;
00230
00231
00232         protected DoubleGroupIterator()
00233         {
00234         }
00235
00236
00237         public DoubleGroupIterator(double[] dSrc, double[] dDest, int nStart, int nStep)
00238             throws IllegalArgumentException
00239         {
00240             super(nStart, (int)dSrc[0], dDest.length, nStep);
00241             if (dSrc.length == 0 || dDest.length == 0 || dSrc.length < dDest.length + 1 || nStart < 0
    || nStep <= 0)
00242                 throw new IllegalArgumentException();
00243             m_dDest = dDest;
00244             m_dSrc = dSrc; // local reference to values
00245         }
00246
00247
00248         @Override
00249         public double[] next()
00250         {
00251             System.arraycopy(m_dSrc, m_nPos, m_dDest, 0, m_dDest.length);
00252             m_nPos += m_nStep; // shift to next group position
00253             return m_dDest;
00254         }
00255     }
00256
00257
00258     private static class IntGroupIterator extends GroupIterator implements Iterator<int[]>
00259     {
00260         private int[] m_nSrc;
00261         private int[] m_nDest;
00262
00263
00264         protected IntGroupIterator()
00265         {
00266         }
00267
00268
00269         public IntGroupIterator(int[] nSrc, int[] nDest, int nStart, int nStep)
00270             throws IllegalArgumentException
00271         {
00272             super(nStart, nSrc[0], nDest.length, nStep);
00273             if (nSrc.length == 0 || nDest.length == 0 || nSrc.length < nDest.length + 1 || nStart < 0
    || nStep <= 0)
00274                 throw new IllegalArgumentException();
00275             m_nDest = nDest;
00276             m_nSrc = nSrc; // local reference to values
00277         }
00278
00279
00280         @Override
00281         public int[] next()
00282         {
00283             System.arraycopy(m_nSrc, m_nPos, m_nDest, 0, m_nDest.length);
00284             m_nPos += m_nStep; // shift to next group position
00285             return m_nDest;
00286         }
00287     }
00288 }
```

8.6 src/cc/util/BufferedInStream.java File Reference

Classes

- class cc.util.BufferedInStream

Packages

- package cc.util

8.7 BufferedInStream.java

Go to the documentation of this file.

```
00001 package cc.util;
00002
00003 import java.io.FilterInputStream;
00004 import java.io.InputStream;
00005 import java.io.IOException;
00006
00007
00008 public class BufferedInStream extends FilterInputStream
00009 {
00010     protected static final int BUFFER_SIZE = 8192;
00011
00012     private int m_nLimit;
00013     private int m_nPos;
00014     private byte[] m_yBuf;
00015
00016
00017     public BufferedInStream(InputStream oInputStream, int nSize)
00018     {
00019         super(oInputStream);
00020         m_yBuf = new byte[nSize];
00021     }
00022
00023
00024     public BufferedInStream(InputStream oInputStream)
00025     {
00026         this(oInputStream, BUFFER_SIZE);
00027     }
00028
00029
00030     @Override
00031     public int read()
00032         throws IOException
00033     {
00034         if (m_nPos >= m_nLimit) // check for empty buffer
00035         {
00036             if ((m_nLimit = in.read(m_yBuf, 0, m_yBuf.length)) <= 0)
00037                 return -1; // no bytes to read and/or read failed
00038
00039             m_nPos = 0; // reset buffer read position
00040         }
00041         return ((int)m_yBuf[m_nPos++]) & 0xff;
00042     }
00043
00044
00045     @Override
00046     public int read(byte[] yBuf, int nOff, int nLen)
00047         throws IOException
00048     {
00049         int nStart = nOff; // save for length calculation
00050         while (nLen > 0) // repeat until request is fulfilled or stream end
00051         {
00052             if (m_nPos >= m_nLimit) // check for empty buffer
00053             {
00054                 if ((m_nLimit = in.read(m_yBuf, 0, m_yBuf.length)) <= 0)
00055                     return nOff - nStart; // no chars to read and/or read failed
00056
00057                 m_nPos = 0; // reset buffer read position
00058             }
00059
00060             int nBytes = Math.min(nLen, m_nLimit - m_nPos); // available bytes
00061             System.arraycopy(m_yBuf, m_nPos, yBuf, nOff, nBytes); // copy buffer
00062             m_nPos += nBytes; // adjust buffer position
00063             nOff += nBytes; // increment dest offset
00064             nLen -= nBytes; // decrement remaining length
00065         }
00066         return nOff - nStart; // return copied byte count
00067     }
00068
00069
00070     @Override
00071     public long skip(long lBytes)
00072         throws IOException
00073     {
```

```
00074         int nAvailable = (m_nLimit - m_nPos);
00075         if (lBytes <= nAvailable)
00076         {
00077             m_nPos += lBytes;
00078             return lBytes;
00079         }
00080 
00081         m_nPos = m_nLimit; // skip buffer entirely
00082         return in.skip(lBytes - nAvailable) + nAvailable; // re-include buffer count
00083     }
00084 }
```

8.8 src/cc/util/CsvReader.java File Reference

Classes

- class cc.util.CsvReader

Packages

- package cc.util

8.9 CsvReader.java

Go to the documentation of this file.

```
00001 package cc.util;
00002 
00003 import java.io.InputStream;
00004 import java.io.IOException;
00005 
00006 
00007 public class CsvReader extends BufferedInStream
00008 {
00009     protected static final int DEFAULT_COLS = 80;
00010 
00011     protected int m_nCol;
00012     protected int[] m_nColEnds;
00013     protected StringBuilder m_sBuf = new StringBuilder(BUFFER_SIZE);
00014     protected char m_cDelim = ',';
00015 
00016     public CsvReader(InputStream oInputStream, int nCols)
00017     {
00018         super(oInputStream);
00019         m_nColEnds = new int[nCols];
00020     }
00021 
00022 
00023     public CsvReader(InputStream oInputStream)
00024     {
00025         this(oInputStream, DEFAULT_COLS);
00026     }
00027 
00028 
00029     public CsvReader(InputStream oInputStream, char cDelim)
00030     {
00031         this(oInputStream, DEFAULT_COLS);
00032         m_cDelim = cDelim;
00033     }
00034 
00035 
00036     public int readLine()
00037         throws IOException
00038     {
00039         m_nCol = 0; // reset column index
00040         m_sBuf.setLength(0); // reset line buffer
00041 
00042         boolean bGo = true;
00043         int nChar;
00044         while (bGo && (nChar = read()) >= 0) // don't advance on line complete
00045         {
```

```
00046             if (nChar == m_cDelim || nChar < ' ')
00047             {
00048                 bGo = (nChar != '\n');
00049                 if (nChar != '\r') // ignore carriage return
00050                     addCol(); // column found
00051             }
00052             else
00053                 m_sBuf.append((char)nChar);
00054         }
00055
00056         if (bGo && m_nCol > 0) // check for missing final newline
00057             addCol();
00058
00059         return m_nCol; // discovered column count
00060     }
00061
00062
00063     private void addCol()
00064     {
00065         if (m_nCol == m_nColEnds.length) // extend column end array
00066         {
00067             int[] nColEnds = new int[m_nCol * 2];
00068             System.arraycopy(m_nColEnds, 0, nColEnds, 0, m_nCol);
00069             m_nColEnds = nColEnds;
00070         }
00071         m_nColEnds[m_nCol++] = m_sBuf.length();
00072     }
00073
00074
00075     private int getStart(int nCol)
00076     {
00077         if (nCol < m_nCol)
00078         {
00079             if (nCol == 0)
00080                 return 0;
00081
00082             return m_nColEnds[nCol - 1];
00083         }
00084         return -1; // force index out of bounds
00085     }
00086
00087
00088     private int getEnd(int nCol)
00089     {
00090         if (nCol < m_nCol)
00091             return m_nColEnds[nCol];
00092
00093         return -1; // force index out of bounds
00094     }
00095
00096
00097     public boolean isNull(int nCol)
00098         throws IndexOutOfBoundsException
00099     {
00100         return (getEnd(nCol) - getStart(nCol) == 0);
00101     }
00102
00103
00104     public double parseDouble(int nCol)
00105         throws IndexOutOfBoundsException, NumberFormatException
00106     {
00107         return Text.parseDouble(m_sBuf, getStart(nCol), getEnd(nCol));
00108     }
00109
00110
00111     public long parseLong(int nCol)
00112         throws IndexOutOfBoundsException, NumberFormatException
00113     {
00114         return Text.parseLong(m_sBuf, getStart(nCol), getEnd(nCol));
00115     }
00116
00117
00118     public float parseFloat(int nCol)
00119         throws IndexOutOfBoundsException, NumberFormatException
00120     {
00121         return (float)parseDouble(nCol);
00122     }
00123
00124
00125     public int parseInt(int nCol)
00126         throws IndexOutOfBoundsException, NumberFormatException
00127     {
00128         return Text.parseInt(m_sBuf, getStart(nCol), getEnd(nCol));
00129     }
00130
00131
00132     public String parseString(int nCol)
```

```
00133         throws IndexOutOfBoundsException
00134     {
00135         return m_sBuf.substring(getStart(nCol), getEnd(nCol));
00136     }
00137
00138
00139     public int parseString(StringBuilder sBuf, int nCol)
00140         throws IndexOutOfBoundsException, NullPointerException
00141     {
00142         sBuf.setLength(0); // clear provided buffer
00143         sBuf.append(m_sBuf, getStart(nCol), getEnd(nCol));
00144         return sBuf.length();
00145     }
00146 }
```

8.10 src/cc/util/Geo.java File Reference

Classes

- class cc.util.Geo

Packages

- package cc.util

8.11 Geo.java

Go to the documentation of this file.

```
00001 package cc.util;
00002
00003 import java.util.Iterator;
00004
00005
00006 public abstract class Geo
00007 {
00008     public final static double EARTH_MINOR_RADIUS = 6356752.0; // in meters
00009     public final static double EARTH_MAJOR_RADIUS = 6378137.0; // in meters
00010     public final static double EARTH_FLATTENING = EARTH_MINOR_RADIUS / EARTH_MAJOR_RADIUS;
00011
00012
00013     private Geo()
00014     {
00015     }
00016
00017
00018     public static int scale(int nVal)
00019     {
00020         return (int)Math.floor(((double)nVal) / 100000.0); // OSM geo-coordinates have 7 decimal
      places
00021     }
00022
00023
00024     public static int getHash(int nLat, int nLon)
00025     {
00026         return (scale(nLon) << 16) + scale(nLat);
00027     }
00028
00029
00030     public static double fromIntDeg(int nOrd)
00031     {
00032         return ((double)nOrd) / 10000000.0;
00033     }
00034
00035
00036     public static int toIntDeg(double dOrd)
00037     {
00038         return (int)(dOrd * 10000000.0);
00039     }
00040
00041
```

```
00042     public static double toMeters(int nOrd)
00043     {
00044         return ((double)nOrd) / 100.0;
00045     }
00046 
00047 
00048     public static double distance(int nXi, int nYi, int nXj, int nYj)
00049     {
00050         return distance(toMeters(nXi), toMeters(nYi), toMeters(nXj), toMeters(nYj));
00051     }
00052 
00053 
00054     public static double distance(double dXi, double dYi, double dXj, double dYj)
00055     {
00056         double dXd = dXj - dXi; // correct distance by latitude
00057 //      dXd = (dXd * Math.cos(Math.toRadians(dYi / 100000.0)) +
00058 //          dXd * Math.cos(Math.toRadians(dYj / 100000.0))) / 2.0;
00059 //      double dYd = (dYj - dYi) * EARTH_FLATTENING;
00060         double dYd = dYj - dYi;
00061         return Math.sqrt(dXd * dXd + dYd * dYd);
00062     }
00063 
00064 
00065 
00066 
00067     public static boolean boundingBoxesIntersect(double dXmin1, double dYmin1, double dXmax1, double
    dYmax1, double dXmin2, double dYmin2, double dXmax2, double dYmax2)
00068     {
00069         return dYmax1 >= dYmin2 && dYmin1 <= dYmax2 && dXmax1 >= dXmin2 && dXmin1 <= dXmax2;
00070     }
00071 
00072 
00086     public static boolean isInside(int nX, int nY,
00087        int nT, int nR, int nB, int nL, int nTol)
00088     {
00089         if (nR < nL) // swap the left and right bounds as needed
00090         {
00091             nR ^= nL;
00092             nL ^= nR;
00093             nR ^= nL;
00094         }
00095 
00096         if (nT < nB) // swap the top and bottom bounds as needed
00097         {
00098             nT ^= nB;
00099             nB ^= nT;
00100             nT ^= nB;
00101         }
00102 
00103         // expand the bounds by the tolerance
00104         return (nX >= nL - nTol && nX <= nR + nTol
00105            && nY >= nB - nTol && nY <= nT + nTol);
00106     }
00107 
00108 
00122     public static boolean isInside(double dX, double dY, double dT, double dR,
00123        double dB, double dL, double dTol)
00124     {
00125         return (dX >= dL - dTol && dX <= dR + dTol
00126            && dY >= dB - dTol && dY <= dT + dTol);
00127     }
00128 
00129 
00130     public static boolean isInBoundingBox(double dX, double dY, double dX1, double dY1, double dX2,
    double dY2)
00131     {
00132         if (dX1 > dX2)
00133         {
00134             double dTemp = dX1;
00135             dX1 = dX2;
00136             dX2 = dTemp;
00137         }
00138 
00139         if (dY1 > dY2)
00140         {
00141             double dTemp = dY1;
00142             dY1 = dY2;
00143             dY2 = dTemp;
00144         }
00145 
00146         return dX >= dX1 && dX <= dX2 && dY >= dY1 && dY <= dY2;
00147     }
00148 
00149 
00150     public static double distAlongLine(double[] dPts, double[] dSeg, double dX, double dY)
00151     {
00152         double dDist = 0.0;
```

```
00153         Iterator<double[]> oIt = Arrays.iterator(dPts, dSeg, 1, 2);
00154         while (oIt.hasNext())
00155         {
00156             oIt.next();
00157             if (isInBoundingBox(dX, dY, dSeg[1], dSeg[0], dSeg[3], dSeg[2]) && collinear(dX, dY,
      dSeg[1], dSeg[0], dSeg[3], dSeg[2]))
00158             {
00159                 dDist += distance(dX, dY, dSeg[1], dSeg[0]);
00160                 return dDist;
00161             }
00162             dDist += distance(dSeg[1], dSeg[0], dSeg[3], dSeg[2]);
00163         }
00164         return Double.NaN;
00165     }
00166 
00167 
00168     public static boolean collinear(double dX1, double dY1, double dX2, double dY2, double dX3, double
      dY3)
00169     {
00170         return Math.abs(dX1 * (dY2 - dY3) + dX2 * (dY3 - dY1) + dX3 * (dY1 - dY2)) < 0.0000001;
00171     }
00172 
00173 
00174     public static double angle(double dX1, double dY1, double dX2, double dY2)
00175     {
00176         return angle(dX1 + 1, dY1, dX1, dY1, dX2, dY2);
00177     }
00178 
00179 
00180     public static double angle(double dX1, double dY1, double dX2, double dY2, double dX3, double dY3)
00181     {
00182         double dUi = dX1 - dX2;
00183         double dUj = dY1 - dY2;
00184         double dVi = dX3 - dX2;
00185         double dVj = dY3 - dY2;
00186         double dDot = dUi * dVi + dUj * dVj;
00187         double dLenU = Math.sqrt(dUi * dUi + dUj * dUj);
00188         double dLenV = Math.sqrt(dVi * dVi + dVj * dVj);
00189         if (dLenU == 0 || dLenV == 0) // prevent division by zero
00190             return Double.NaN;
00191         double dValue = dDot / (dLenU * dLenV);
00192         dValue = (dValue * 1000000000000L) / 1000000000000L; // round to help prevent value outside of
      the domain of arccos
00193         if (dValue > 1 || dValue < -1) // prevent domain error for arcos
00194             return Double.NaN;
00195 
00196         return Math.acos(dValue); // return value in radians
00197     }
00198 }
```

8.12 src/cc/util/MathUtil.java File Reference

Classes

- class cc.util.MathUtil

Packages

- package cc.util

8.13 MathUtil.java

Go to the documentation of this file.

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005  */
00006 package cc.util;
00007 
```

```
00012 public abstract class MathUtil
00013 {
00014     public static double TWOPI = Math.PI * 2;
00015
00016
00017     public static void getIntersection(double dPx, double dPy, double dEnd1x, double dEnd1y, double
      dQx, double dQy, double dEnd2x, double dEnd2y, double[] dInter)
00018     {
00019         dInter[0] = Double.NaN;
00020         dInter[1] = Double.NaN;
00021         double dDeltaQPx = dQx - dPx;
00022         double dDeltaQPy = dQy - dPy;
00023         double dRx = dEnd1x - dPx;
00024         double dRy = dEnd1y - dPy;
00025         double dSx = dEnd2x - dQx;
00026         double dSy = dEnd2y - dQy;
00027         double dRCrossS = cross(dRx, dRy, dSx, dSy);
00028         if (dRCrossS == 0)
00029             return;
00030         double dT = cross(dDeltaQPx, dDeltaQPy, dSx, dSy) / dRCrossS;
00031         if (dT < 0 || dT > 1)
00032             return;
00033         double dU = cross(dDeltaQPx, dDeltaQPy, dRx, dRy) / dRCrossS;
00034         if (dU < 0 || dU > 1)
00035             return;
00036         dInter[0] = dPx + dT * dRx;
00037         dInter[1] = dPy + dT * dRy;
00038     }
00039
00040
00041     public static double cross(double dVx, double dVy, double dWx, double dWy)
00042     {
00043         return dVx * dWy - dVy * dWx;
00044     }
00045
00046
00047     public static int compareTol(double d1, double d2, double dTol)
00048     {
00049         if (d2 > d1)
00050         {
00051             if (d2 - d1 > dTol)
00052                 return -1;
00053         }
00054         else if (d1 - d2 > dTol)
00055             return 1;
00056         return 0;
00057     }
00058
00059
00060     public static double cubic(double dX, double dA, double dB, double dC, double dD)
00061     {
00062         return dA + (dB * dX) + (dC * dX * dX) + (dD * dX * dX * dX);
00063     }
00064
00065
00066     public static double normalizeRadians(double dRad)
00067     {
00068         if (dRad < 0)
00069             return dRad + Math.ceil(-dRad / TWOPI) * TWOPI;
00070         else if (dRad >= TWOPI)
00071             return dRad - Math.floor(dRad / TWOPI) * TWOPI;
00072
00073         return dRad;
00074     }
00075 }
```

8.14 src/cc/util/StringPool.java File Reference

Classes

- class cc.util.StringPool
- class cc.util.StringPool.Group

Packages

- package cc.util

8.15 StringPool.java

Go to the documentation of this file.

```
00001 package cc.util;
00002
00003 import java.util.ArrayList;
00004 import java.util.Collections;
00005
00006
00007 public class StringPool extends ArrayList<StringPool.Group>
00008 {
00009     protected char[] m_oSearch = new char[2];
00010
00011
00012     public StringPool()
00013     {
00014     }
00015
00016
00017     public String intern(String sVal)
00018     {
00019         int nIndex = m_oSearch.length;
00020         while (nIndex-- > 0) // create search key
00021         {
00022             if (nIndex < sVal.length())
00023                 m_oSearch[nIndex] = Character.toUpperCase(sVal.charAt(nIndex));
00024             else
00025                 m_oSearch[nIndex] = 0;
00026         }
00027
00028         nIndex = Collections.binarySearch(this, m_oSearch);
00029         if (nIndex < 0) // completely new string group array
00030         {
00031             nIndex = ~nIndex;
00032             Group oGroup = new Group(m_oSearch);
00033             add(nIndex, oGroup);
00034         }
00035
00036         Group oGroup = get(nIndex);
00037         nIndex = Collections.binarySearch(oGroup, sVal);
00038         if (nIndex < 0)
00039         {
00040             oGroup.add(~nIndex, sVal);
00041             return sVal;
00042         }
00043         return oGroup.get(nIndex);
00044     }
00045
00046
00047     public ArrayList<String> toList()
00048     {
00049         int nSize = 0; // determine space requirements
00050         for (Group oGroup : this)
00051             nSize += oGroup.size();
00052
00053         ArrayList<String> oList = new ArrayList(nSize);
00054         for (Group oGroup : this)
00055             oList.addAll(oGroup);
00056
00057         Collections.sort(oList); // correct pool grouping order
00058         return oList;
00059     }
00060
00061
00062     @Override
00063     public void clear()
00064     {
00065         for (Group oGroup : this)
00066             oGroup.clear();
00067
00068         super.clear();
00069     }
00070
00071
00072     class Group extends ArrayList<String> implements Comparable<char[]>
00073     {
00074         char[] m_oKey;
00075
00076
00077         private Group()
00078         {
00079         }
00080
00081
00082         Group(char[] oKey)
```

```
00083         {
00084             m_oKey = new char[]{oKey[0], oKey[1]};
00085         }
00086 
00087 
00088         @Override
00089         public int compareTo(char[] oRhs)
00090         {
00091             int nComp = m_oKey[0] - oRhs[0];
00092             if (nComp == 0)
00093                 nComp = m_oKey[1] - oRhs[1];
00094 
00095             return nComp;
00096         }
00097     }
00098 }
```

8.16 src/cc/util/Text.java File Reference

Classes

- class cc.util.Text

Packages

- package cc.util

8.17 Text.java

Go to the documentation of this file.

```
00001 package cc.util;
00002 
00003 import java.nio.ByteBuffer;
00004 import java.util.Base64;
00005 import java.util.UUID;
00006 
00007 
00012 public abstract class Text
00013 {
00017     private static final char[] HEX_CHARS =
00018             {
00019                     '0', '1', '2', '3', '4', '5', '6', '7',
00020                     '8', '9', 'a', 'b', 'c', 'd', 'e', 'f'
00021             };
00025     private static final int DIGIT_OFFSET = 48;
00029     private static final int MIN_EXPONENT = -323;
00033     private static final int MAX_EXPONENT = 308;
00034 
00038     private static final int INIT_CHAR = 0;
00042     private static final int DECIMAL = 1;
00046     private static final int FRACTION = 2;
00050     private static final int EXPONENT = 3;
00054     private static final int PARSE_END = 4;
00055 
00059     private static final String POS_INF = "\u221E";
00063     private static final String NEG_INF = "-\u221E";
00068     private static final String POS_INFINITY = "Infinity";
00073     private static final String NEG_INFINITY = "-Infinity";
00078     private static final String NAN = "NaN";
00082     public static final Base64.Encoder B64ENC = Base64.getUrlEncoder().withoutPadding();
00083 
00084     private static final ByteBuffer UUID_BUFFER = ByteBuffer.allocate(16);
00085 
00086 
00093     private Text()
00094         {
00095     }
00096 
00097 
00103     public static void removeWhitespace(StringBuilder sBuffer) {
```

```
00104         // only remove whitespace if there is something to scan
00105         if (sBuffer.length() > 0) {
00106             // reverse iterate to the first non-whitespace character
00107             int nIndex = sBuffer.length();
00108             while (nIndex-- > 0 &&
00109                     Character.isWhitespace(sBuffer.charAt(nIndex))) ;
00110
00111             // remove the trailing whitespace segment
00112             sBuffer.delete(++nIndex, sBuffer.length());
00113
00114             // forward iterate to the first non-whitespace character
00115             nIndex = 0;
00116             while (nIndex < sBuffer.length() &&
00117                     Character.isWhitespace(sBuffer.charAt(nIndex++))) ;
00118
00119             // remove the leading whitespace segment
00120             if (nIndex > 0)
00121                 sBuffer.delete(0, --nIndex);
00122         }
00123     }
00124
00125
00133     public static long parseLong(CharSequence iCharSeq) {
00134         return parseLong(iCharSeq, 0, iCharSeq.length());
00135     }
00136
00137
00151     public static long parseLong(CharSequence iCharSeq, int nPos, int nEndPos)
00152             throws NumberFormatException {
00153         // test for the sign character
00154         boolean bNegative = (iCharSeq.charAt(nPos) == '-');
00155         if (bNegative)
00156             ++nPos;
00157
00158         int nDigit = 0;
00159         long lValue = 0L;
00160         while (nPos < nEndPos) {
00161             // map the character value to the numeric value
00162             nDigit = iCharSeq.charAt(nPos++) - DIGIT_OFFSET;
00163             // test for characters that are not numbers
00164             if (nDigit < 0 || nDigit > 9)
00165                 throw new NumberFormatException();
00166
00167             // shift the existing value and add the new digit value
00168             lValue *= 10L;
00169             lValue += nDigit;
00170         }
00171
00172         if (bNegative)
00173             return -lValue;
00174
00175         return lValue;
00176     }
00177
00178
00186     public static int parseInt(CharSequence iCharSeq) {
00187         return parseInt(iCharSeq, 0, iCharSeq.length());
00188     }
00189
00190
00204     public static int parseInt(CharSequence iCharSeq, int nPos, int nEndPos)
00205             throws NumberFormatException {
00206         int nSign = 1;
00207         int nValue = 0;
00208
00209         // valid characters for an integer are -, +, and digits
00210         int nState = INIT_CHAR;
00211         while (nPos < nEndPos && nState != PARSE_END) {
00212             char cDigit = iCharSeq.charAt(nPos++);
00213             switch (nState) {
00214                 // the digits test is first since it is the most likely
00215                 case DECIMAL: {
00216                     if (Character.isDigit(cDigit)) {
00217                         // shift any existing value
00218                         nValue *= 10;
00219                         nValue += (cDigit - DIGIT_OFFSET);
00220                     } else
00221                         throw new NumberFormatException("Illegal character '" +
00222                                 iCharSeq.charAt(--nPos) + "' for integer " +
00223                                 "expression at position " + nPos + ".");
00224                 }
00225                 break;
00226
00227                 // the initial character test is only performed once
00228                 case INIT_CHAR: {
00229                     nState = DECIMAL;
00230                     if (cDigit == '-')
```

```
00231                       nSign = -1;
00232                   else
00233                       // back up one character to test for digits
00234                       --nPos;
00235               }
00236           }
00237       }
00238
00239       return (nSign * nValue);
00240   }
00241
00242
00243   public static double parseDouble(CharSequence iCharSeq, int nPos, int nEndPos)
00244       throws IndexOutOfBoundsException, NumberFormatException
00245   {
00246       boolean bNeg = (iCharSeq.charAt(nPos) == '-');
00247       if (bNeg) // test for sign character
00248           ++nPos;
00249
00250       double dVal = 0.0;
00251       while (nPos < nEndPos && iCharSeq.charAt(nPos) != '.')
00252       {
00253           int nDigit = iCharSeq.charAt(nPos++) - DIGIT_OFFSET; // map char value
00254           if (nDigit < 0 || nDigit > 9) // check for non-numeric chars
00255               throw new NumberFormatException();
00256
00257           dVal *= 10.0; // shift existing value
00258           dVal += nDigit; // add new digit value
00259       }
00260
00261       if (iCharSeq.charAt(nPos++) == '.') // fractional part check
00262       {
00263           double dDiv = 1.0;
00264           double dFrac = 0.0;
00265           while (nPos < nEndPos)
00266           {
00267               int nDigit = iCharSeq.charAt(nPos++) - DIGIT_OFFSET; // map char value
00268               if (nDigit < 0 || nDigit > 9) // check for non-numeric chars
00269                   throw new NumberFormatException();
00270
00271               dDiv *= 10.0; // track decimal places
00272               dFrac *= 10.0; // shift existing value
00273               dFrac += nDigit; // add new digit value
00274           }
00275           dVal += dFrac / dDiv; // expensive division operator
00276       }
00277
00278       if (bNeg)
00279           return -dVal;
00280
00281       return dVal;
00282   }
00283
00284
00291   public static double parseDouble(CharSequence iCharSeq) {
00292       // first test for expected string names
00293       if (compare(iCharSeq, POS_INF) == 0 ||
00294               compare(iCharSeq, POS_INFINITY) == 0)
00295           return Double.POSITIVE_INFINITY;
00296
00297       if (compare(iCharSeq, NEG_INF) == 0 ||
00298               compare(iCharSeq, NEG_INFINITY) == 0)
00299           return Double.NEGATIVE_INFINITY;
00300
00301       if (compare(iCharSeq, NAN) == 0)
00302           return Double.NaN;
00303
00304       int nExponent = 0;
00305       double dSign = 1.0;
00306       double dValue = 0.0;
00307       double dMultiplier = 0.1;
00308       double dFraction = 0.0;
00309
00310       // valid characters for a double are -, +, ., digits, E, and e
00311       int nIndex = 0;
00312       int nState = INIT_CHAR;
00313       while (nIndex < iCharSeq.length() && nState != PARSE_END) {
00314           char cDigit = iCharSeq.charAt(nIndex++);
00315           switch (nState) {
00316               // the digits test is first since it is the most likely
00317               // parseInt cannot be used due to potential leading zeros
00318               // in the decimal and fractional parts, i.e. -0.00763
00319               case DECIMAL: {
00320                   if (Character.isDigit(cDigit)) {
00321                       // shift any existing value
00322                       dValue *= 10.0;
00323                       dValue += (cDigit - DIGIT_OFFSET);
```

```
00324                     } else {
00325                         // switch to other states for the double interpretation
00326                         switch (cDigit) {
00327                             case '.':
00328                                 nState = FRACTION;
00329                                 break;
00330 
00331                             case 'e':
00332                             case 'E':
00333                                 nState = EXPONENT;
00334                                 break;
00335 
00336                             default:
00337                                 nState = PARSE_END;
00338                                 break;
00339                         }
00340                     }
00341                 }
00342                 break;
00343 
00344                 case FRACTION: {
00345                     if (Character.isDigit(cDigit)) {
00346                         dFraction += (cDigit - DIGIT_OFFSET) * dMultiplier;
00347                         dMultiplier *= 0.1;
00348                     } else if (cDigit == 'e' || cDigit == 'E')
00349                         nState = EXPONENT;
00350                     else
00351                         nState = PARSE_END;
00352                 }
00353                 break;
00354 
00355                 // the exponent state is able to use the parseInt method
00356                 // as any fractional exponent will be ignored
00357                 case EXPONENT: {
00358                     nExponent = parseInt(iCharSeq, --nIndex, iCharSeq.length());
00359                     nState = PARSE_END;
00360                 }
00361                 break;
00362 
00363                 // the initial character test is only performed once
00364                 case INIT_CHAR: {
00365                     switch (cDigit) {
00366                         case '-': {
00367                             dSign = -1.0;
00368                             nState = DECIMAL;
00369                         }
00370                         break;
00371 
00372                         case '+': {
00373                             dSign = 1.0;
00374                             nState = DECIMAL;
00375                         }
00376                         break;
00377 
00378                         case '.':
00379                             nState = FRACTION;
00380                             break;
00381 
00382                         default: {
00383                             if (Character.isDigit(cDigit)) {
00384                                 // back up one character to test for digits
00385                                 --nIndex;
00386                                 nState = DECIMAL;
00387                             } else
00388                                 nState = PARSE_END;
00389                         }
00390                     }
00391                 }
00392             }
00393         }
00394 
00395         // only generate an exponent when necessary
00396         double dExponent = 1.0;
00397         if (nExponent != 0) {
00398             // check the limits of the exponent
00399             if (nExponent < MIN_EXPONENT)
00400                 nExponent = MIN_EXPONENT;
00401 
00402             if (nExponent > MAX_EXPONENT)
00403                 nExponent = MAX_EXPONENT;
00404 
00405             dExponent = Math.pow(10.0, (double) nExponent);
00406         }
00407 
00408         return (dSign * (dValue + dFraction) * dExponent);
00409     }
00410 
```

```
00411
00425     public static int compare(CharSequence iSeqL, CharSequence iSeqR) {
00426         int nCompare = 0;
00427         int nIndex = -1;
00428         int nLimit = Math.min(iSeqL.length(), iSeqR.length());
00429
00430         while (nCompare == 0 && ++nIndex < nLimit)
00431             nCompare = (iSeqL.charAt(nIndex) - iSeqR.charAt(nIndex));
00432
00433         if (nCompare == 0)
00434             nCompare = (iSeqL.length() - iSeqR.length());
00435
00436         return nCompare;
00437     }
00438
00439
00449     public static int compareIgnoreCase(CharSequence iSeqL, CharSequence iSeqR) {
00450         int nCompare = 0;
00451         int nIndex = -1;
00452         int nLimit = Math.min(iSeqL.length(), iSeqR.length());
00453
00454         while (nCompare == 0 && ++nIndex < nLimit)
00455             nCompare =
00456                     (
00457                             Character.toLowerCase(iSeqL.charAt(nIndex)) -
00458                                     Character.toLowerCase(iSeqR.charAt(nIndex))
00459                     );
00460
00461         if (nCompare == 0)
00462             nCompare = (iSeqL.length() - iSeqR.length());
00463
00464         return nCompare;
00465     }
00466
00467
00477     public static boolean startsWith(CharSequence iSource, CharSequence iPrefix) {
00478         int nIndex = iPrefix.length();
00479
00480         // the source cannot start with a pattern with more characters
00481         if (nIndex > iSource.length())
00482             return false;
00483
00484         // compare each characters starting at the end of the sequence
00485         boolean bMatch = true;
00486         while (bMatch && nIndex-- > 0)
00487             bMatch = (iSource.charAt(nIndex) == iPrefix.charAt(nIndex));
00488
00489         return bMatch;
00490     }
00491
00492
00502     public static boolean endsWith(CharSequence iSource, CharSequence iSuffix) {
00503         int nIndex = iSuffix.length();
00504         int nSrcIndex = iSource.length();
00505
00506         // the source cannot end with a pattern with more characters
00507         if (nIndex > nSrcIndex)
00508             return false;
00509
00510         // compare each characters starting at the end of the sequence
00511         boolean bMatch = true;
00512         while (bMatch && nIndex-- > 0)
00513             bMatch = (iSource.charAt(--nSrcIndex) == iSuffix.charAt(nIndex));
00514
00515         return bMatch;
00516     }
00517
00518
00525     public static String toHexString(byte[] yBytes)
00526         {
00527             return toHexString(yBytes, 0, yBytes.length);
00528     }
00529
00530
00537     public static void toHexString(byte[] yBytes, StringBuilder sBuf)
00538         {
00539             toHexString(yBytes, 0, yBytes.length, sBuf);
00540     }
00541
00542
00551     public static String toHexString(byte[] yBytes, int nOffset, int nLength)
00552         {
00553             StringBuilder sBuf = new StringBuilder();
00554             toHexString(yBytes, nOffset, nLength, sBuf);
00555             return sBuf.toString();
00556         }
00557
```

```
00558
00567     public static void toHexString(byte[] yBytes, int nOffset,
00568             int nLength, StringBuilder sBuf)
00569         {
00570             for (; nOffset < nLength; nOffset++)
00571             {
00572                 sBuf.append(HEX_CHARS[((yBytes[nOffset] & 0xf0) >> 4)]);
00573                 sBuf.append(HEX_CHARS[(yBytes[nOffset] & 0x0f)]);
00574             }
00575     }
00576
00577
00584     public static byte[] fromHexString(StringBuilder sBuf)
00585         {
00586             if (sBuf == null || sBuf.length() == 0)
00587                 return null;
00588
00589             if (sBuf.length() % 2 != 0)
00590                 sBuf.append("0");
00591
00592             byte[] yBytes = new byte[sBuf.length() / 2];
00593             for (int nIndex = 0; nIndex < yBytes.length; nIndex++)
00594             {
00595                 int nPos = nIndex * 2;
00596                 yBytes[nIndex] = (byte)((Character.digit(sBuf.charAt(nPos), 16) << 4) +
00597                     Character.digit(sBuf.charAt(nPos + 1), 16));
00598             }
00599             return yBytes;
00600     }
00601
00602
00611     public static void replaceAll(StringBuilder sBuffer,
00612                                   String sSearch, String sReplace) {
00613         int nIndex = 0;
00614         while ((nIndex = sBuffer.indexOf(sSearch, nIndex)) >= 0) {
00615             sBuffer.replace(nIndex, nIndex + sSearch.length(), sReplace);
00616             // move start point after the replacement to enable recursion
00617             nIndex += sReplace.length();
00618         }
00619     }
00620
00621
00631     public static int getBytes(byte[] yBuffer, CharSequence iCharSeq) {
00632         // determine the maximum the available capacity
00633         int nLength = iCharSeq.length();
00634         if (nLength > yBuffer.length)
00635             nLength = yBuffer.length;
00636
00637         for (int nIndex = 0; nIndex < nLength; nIndex++)
00638             yBuffer[nIndex] = (byte) iCharSeq.charAt(nIndex);
00639
00640         return nLength;
00641     }
00642
00643
00652     public static String truncate(String sValue, int nLength) {
00653         if(sValue == null || sValue.length() <= nLength)
00654           return sValue;
00655         else
00656           return sValue.substring(0, nLength);
00657     }
00658
00659
00660     public static String getUUID()
00661     {
00662         UUID oUuid = UUID.randomUUID();
00663         synchronized(UUID_BUFFER) // shared byte buffer
00664         {
00665             UUID_BUFFER.clear();
00666             UUID_BUFFER.putLong(oUuid.getMostSignificantBits());
00667             UUID_BUFFER.putLong(oUuid.getLeastSignificantBits());
00668             return B64ENC.encodeToString(UUID_BUFFER.array());
00669         }
00670     }
00671 }
```

8.18 src/cc/ws/EventMgr.java File Reference

Classes

- class cc.ws.EventMgr

Packages

- package cc.ws

8.19 EventMgr.java

Go to the documentation of this file.

```
00001 package cc.ws;
00002
00003 import cc.util.Arrays;
00004 import java.io.BufferedWriter;
00005 import java.io.FileInputStream;
00006 import java.io.FileWriter;
00007 import java.util.ArrayList;
00008 import java.util.Collections;
00009 import java.util.HashMap;
00010 import javax.servlet.http.HttpServletRequest;
00011 import cc.util.CsvReader;
00012 import cc.util.Text;
00013 import java.io.IOException;
00014 import java.io.PrintWriter;
00015 import java.util.Iterator;
00016
00017
00018 public class EventMgr extends Handler implements Runnable
00019 {
00020     private final ArrayList<String[]> m_oEvents = new ArrayList();
00021     private final HashMap<String, String> m_oTypes = new HashMap();
00022     private final HashMap<String, String> m_oLanes = new HashMap();
00023     private double m_dTol = 0.00001;
00024
00025     String m_sEventFile; // path to file containing event detail data
00026
00027
00028     public EventMgr()
00029     {
00030         m_oTypes.put("1", "work zone");
00031         m_oTypes.put("2", "incident");
00032
00033         m_oLanes.put("8193", "all");
00034         m_oLanes.put("8195", "left-lane");
00035         m_oLanes.put("8196", "right-lane");
00036         m_oLanes.put("81952", "left-2-lanes");
00037         m_oLanes.put("81953", "left-3-lanes");
00038         m_oLanes.put("81962", "right-2-lanes");
00039         m_oLanes.put("81963", "right-3-lanes");
00040         m_oLanes.put("8197", "center");
00041         m_oLanes.put("8198", "middle-lane");
00042         m_oLanes.put("8200", "right-turning-lane");
00043         m_oLanes.put("8201", "left-turning-lane");
00044         m_oLanes.put("8239", "right-exit-lane");
00045         m_oLanes.put("8240", "left-exit-lane");
00046         m_oLanes.put("8241", "right-merging-lane");
00047         m_oLanes.put("8242", "left-merging-lane");
00048         m_oLanes.put("8202", "right-exit-ramp");
00049         m_oLanes.put("8243", "right-second-exit-ramp");
00050         m_oLanes.put("8203", "right-entrance-ramp");
00051         m_oLanes.put("8245", "right-second-entrance-ramp");
00052         m_oLanes.put("8204", "left-exit-ramp");
00053         m_oLanes.put("8244", "left-second-exit-ramp");
00054         m_oLanes.put("8205", "left-entrance-ramp");
00055         m_oLanes.put("8246", "left-second-entrance-ramp");
00056         m_oLanes.put("82281", "sidewalk");
00057         m_oLanes.put("8228", "bike-lane");
00058         m_oLanes.put("0", "none");
00059         m_oLanes.put("8247", "unknown");
00060         m_oLanes.put("81980", "alternate-flow-lane");
00061         m_oLanes.put("81951", "shift-left");
00062         m_oLanes.put("81961", "shift-right");
00063     }
00064
00065
00066     @Override
00067     public void init()
00068     {
00069         String sEventFile = getServletConfig().getInitParameter("eventfile");
00070         if (sEventFile != null && sEventFile.length() > 0)
00071         {
00072             m_sEventFile = sEventFile;
00073             new Thread(this).start();
```

```
00074         }
00075         String sTol = getServletConfig().getInitParameter("tol");
00076         if (sTol != null && sTol.length() > 0)
00077             m_dTol = Double.parseDouble(sTol);
00078     }
00079 
00080 
00081     @Override
00082     public void run()
00083     {
00084         try (CsvReader oIn = new CsvReader(new FileInputStream(m_sEventFile), '\t'))
00085         {
00086             int nCols;
00087             String[] sSearch = new String[1];
00088             synchronized(m_oEvents)
00089             {
00090                 while ((nCols = oIn.readLine()) > 0)
00091                 {
00092                     sSearch[0] = oIn.parseString(0);
00093                     int nIndex = Collections.binarySearch(m_oEvents, sSearch, STR_ARR_COMP); // search
      for id in list
00094                     if (nIndex < 0)
00095                     {
00096                         nIndex = ~nIndex;
00097                         m_oEvents.add(nIndex, new String[nCols]); // right now we have
      uuid,user,timestamp,status,JSON
00098                     }
00099                     update(m_oEvents.get(nIndex), nCols, oIn); // replace with most recent update
00100                 }
00101             }
00102         }
00103         catch (Exception oEx)
00104         {
00105         }
00106     }
00107 
00108 
00109     protected void doNull(Session oSess, HttpServletRequest oReq, PrintWriter oOut)
00110     {
00111         oOut.write("{");
00112         synchronized(m_oEvents)
00113         {
00114             int nCount = 0;
00115             for (String[] sEvent : m_oEvents)
00116             {
00117                 String sData = sEvent[sEvent.length - 1];
00118                 int nStart = sData.indexOf("\"pts\":[");
00119                 if (nStart < 0)
00120                     continue;
00121 
00122                 nStart += "\"pts\":[".length();
00123                 int nEnd = sData.indexOf("]", nStart);
00124                 String[] sOrdinates = sData.substring(nStart, nEnd).split(",");
00125                 if (nCount++ > 0)
00126                     oOut.write(",");
00127 
00128                 oOut.write(String.format("\"%s\":[", sEvent[0]));
00129 
00130                 int nSize = sOrdinates.length;
00131                 if (nSize > 0)
00132                 {
00133                     oOut.write(sOrdinates[0]);
00134                     for (int i = 1; i < nSize; i++)
00135                     {
00136                         oOut.write(",");
00137                         oOut.write(sOrdinates[i]); // {"uuid" : pt array, "uuid" : pt array, ...}
00138                     }
00139                 }
00140                 oOut.write("]");
00141             }
00142         }
00143         oOut.write("}");
00144     }
00145 
00146 
00147     protected void doList(Session oSess, HttpServletRequest oReq, PrintWriter oOut)
00148     {
00149         double dLat1;
00150         double dLat2;
00151         double dLon1;
00152         double dLon2;
00153         try
00154         {
00155             dLat1 = Double.parseDouble(oReq.getParameter("lat1")); // filter by selected point
00156             dLat2 = Double.parseDouble(oReq.getParameter("lat2"));
00157             if (dLat1 > dLat2)
00158             {
```

```
00159                 double dTemp = dLat2;
00160                 dLat2 = dLat1;
00161                 dLat1 = dTemp;
00162             }
00163 
00164             dLon1 = Double.parseDouble(oReq.getParameter("lon1"));
00165             dLon2 = Double.parseDouble(oReq.getParameter("lon2"));
00166             if (dLon1 > dLon2)
00167             {
00168                 double dTemp = dLon2;
00169                 dLon2 = dLon1;
00170                 dLon1 = dTemp;
00171             }
00172         }
00173         catch (Exception oEx)
00174         {
00175             oOut.write("{}"); // send empty object if error getting parameters
00176             return;
00177         }
00178 
00179         oOut.write("{");
00180         synchronized(m_oEvents)
00181         {
00182             double[] dPoints = Arrays.newDoubleArray();
00183             double[] dSeg = new double[4];
00184             int nCount = 0;
00185             for (String[] sEvent : m_oEvents)
00186             {
00187                 dPoints[0] = 1;
00188                 String sData = sEvent[sEvent.length - 1];
00189                 int nStart = sData.indexOf("\"pts\":[");
00190                 if (nStart < 0)
00191                     continue;
00192 
00193                 nStart += "\"pts\":[".length();
00194                 int nEnd = sData.indexOf("]", nStart);
00195                 String[] sOrdinates = sData.substring(nStart, nEnd).split(",");
00196                 for (int i = 0; i < sOrdinates.length;)
00197                 {
00198                     dPoints = Arrays.add(dPoints,
00199                        Math.round(Double.parseDouble(sOrdinates[i++]) * 10000000.0 + 0.000001) /
    10000000.0,
00200                        Math.round(Double.parseDouble(sOrdinates[i++]) * 10000000.0 + 0.000001) /
    10000000.0);
00201                 }
00202 
00203                 Iterator<double[]> oIt = Arrays.iterator(dPoints, dSeg, 1, 2);
00204                 while (oIt.hasNext())
00205                 {
00206                     oIt.next();
00207                     double dSegLatMin = dSeg[0];
00208                     double dSegLatMax = dSeg[2];
00209                     double dSegLonMin = dSeg[1];
00210                     double dSegLonMax = dSeg[3];
00211 
00212                     if (dSegLatMin > dSegLatMax)
00213                     {
00214                         double dTemp = dSegLatMax;
00215                         dSegLatMax = dSegLatMin;
00216                         dSegLatMin = dTemp;
00217                     }
00218 
00219                     if (dSegLonMin > dSegLonMax)
00220                     {
00221                         double dTemp = dSegLonMax;
00222                         dSegLonMax = dSegLonMin;
00223                         dSegLonMin = dTemp;
00224                     }
00225 
00226                     dSegLatMin -= m_dTol;
00227                     dSegLatMax += m_dTol;
00228                     dSegLonMin -= m_dTol;
00229                     dSegLonMax += m_dTol;
00230 
00231                     if (dLon2 < dSegLonMin || dLon1 > dSegLonMax ||
00232                         dLat2 < dSegLatMin || dLat1 > dSegLatMax)
00233                         continue;
00234 
00235                     if (nCount++ > 0)
00236                         oOut.write(',');
00237                     writeJson(oOut, sEvent[0], sData);
00238 
00239                     break;
00240                 }
00241             }
00242         }
00243         oOut.write("}");
```

```
00244     }
00245
00246
00247     protected void doDetail(Session oSess, HttpServletRequest oReq, PrintWriter oOut)
00248     {
00249         String sEventId = oReq.getParameter("id");
00250         if (sEventId == null || m_oEvents.isEmpty())
00251             return;
00252
00253         synchronized(m_oEvents) // ensure any changes are committed
00254         {
00255             String[] sSearch = new String[]{sEventId};
00256             int nIndex = Collections.binarySearch(m_oEvents, sSearch, STR_ARR_COMP);
00257             if (nIndex < 0) // list only contains most recent updates
00258             {
00259                 oOut.write("{}");
00260                 return;
00261             }
00262
00263             String[] sEvent = m_oEvents.get(nIndex);
00264             oOut.write("{");
00265             writeJson(oOut, sEvent[0], sEvent[sEvent.length - 1]);
00266             oOut.write("}");
00267         }
00268     }
00269
00270
00271     protected void doSave(Session oSess, HttpServletRequest oReq, PrintWriter oOut)
00272        throws IOException
00273     {
00274         String sEventId = oReq.getParameter("id");
00275         if (sEventId == null || sEventId.length() == 0) // new event condition
00276             sEventId = Text.getUUID();
00277
00278         synchronized(m_oEvents)
00279         {
00280             String sUsername = oSess.m_oUser.m_sUser; // set event parameters
00281
00282             String[] sSearch = new String[]{sEventId};
00283             int nIndex = Collections.binarySearch(m_oEvents, sSearch, STR_ARR_COMP);
00284             if (nIndex < 0)
00285             {
00286                 nIndex = ~nIndex;
00287                 m_oEvents.add(nIndex, new String[]{sEventId, sUsername, null,
    oReq.getParameter("data")}); // insert new event
00288             }
00289             String[] sEvent = m_oEvents.get(nIndex);
00290             synchronized (ISO8601Sdf)
00291             {
00292                 sEvent[2] = ISO8601Sdf.format(System.currentTimeMillis()); // set update time last
00293             }
00294
00295             try (BufferedWriter oFileOut = new BufferedWriter(new FileWriter(m_sEventFile, true)))
00296             {
00297                 oFileOut.write(sEvent[0]);
00298                 for (int i = 1; i < sEvent.length; i++)
00299                 {
00300                     oFileOut.write("\t");
00301                     oFileOut.write(sEvent[i]);
00302                 }
00303                 oFileOut.write("\n");
00304             }
00305             oOut.write("{");
00306             writeJson(oOut, sEvent[0], sEvent[sEvent.length - 1]); // id:data
00307             oOut.write("}");
00308         }
00309     }
00310
00311
00312     protected void doTypes(Session oSess, HttpServletRequest oReq, PrintWriter oOut)
00313     {
00314         oOut.write("{");
00315         if (m_oTypes.size() > 0)
00316         {
00317             StringBuilder sBuffer = new StringBuilder();
00318             m_oTypes.entrySet().forEach(oEntry -> sBuffer.append(String.format("\"%s\":\"%s\",",
    oEntry.getKey(), oEntry.getValue())));
00319             sBuffer.setLength(sBuffer.length() - 1); // remove last comma
00320             oOut.write(sBuffer.toString());
00321         }
00322         oOut.write("}");
00323     }
00324
00325
00326     protected void doLanes(Session oSess, HttpServletRequest oReq, PrintWriter oOut)
00327     {
00328         oOut.write("{");
```

```
00329         if (m_oLanes.size() > 0)
00330         {
00331             StringBuilder sBuffer = new StringBuilder();
00332             m_oLanes.entrySet().forEach(oEntry -> sBuffer.append(String.format("\"%s\":\"%s\",",
      oEntry.getKey(), oEntry.getValue())));
00333             sBuffer.setLength(sBuffer.length() - 1); // remove last comma
00334             oOut.write(sBuffer.toString());
00335         }
00336         oOut.write("}");
00337     }
00338 }
```

8.20 src/cc/ws/Handler.java File Reference

Classes

- class cc.ws.Handler

Packages

- package cc.ws

8.21 Handler.java

Go to the documentation of this file.

```
00001 package cc.ws;
00002
00003 import cc.util.CsvReader;
00004 import cc.util.Text;
00005 import java.io.IOException;
00006 import java.io.PrintWriter;
00007 import java.lang.reflect.Method;
00008 import java.text.SimpleDateFormat;
00009 import java.util.Comparator;
00010 import java.util.SimpleTimeZone;
00011 import javax.servlet.http.HttpServletRequest;
00012 import javax.servlet.ServletException;
00013 import javax.servlet.http.HttpServlet;
00014 import javax.servlet.http.HttpServletResponse;
00015
00016
00017 public class Handler extends HttpServlet
00018 {
00019     protected static Comparator<String[]> STR_ARR_COMP = (String[] o1, String[] o2) ->
      o1[0].compareTo(o2[0]);
00020     protected final static SimpleDateFormat ISO8601Sdf = new
      SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss'Z'");
00021
00022     static
00023     {
00024         ISO8601Sdf.setTimeZone(new SimpleTimeZone(0, "")); // set utc time
00025     }
00026
00027     @Override
00028     protected void doPost(HttpServletRequest oReq, HttpServletResponse oRep)
00029        throws ServletException, IOException
00030     {
00031         try
00032         {
00033             Session oSess = SessMgr.getSession(oReq);
00034             if (oSess == null)
00035             {
00036                 oRep.sendError(401);
00037                 return;
00038             }
00039
00040
00041             StringBuilder sMethod = new StringBuilder("do");
00042             String sAction = oReq.getPathInfo();
00043             sMethod.append(sAction);
```

```
00044             if (sMethod.charAt(2) == '/') // remove leading "/"
00045                 sMethod.deleteCharAt(2);
00046 
00047             sMethod.setCharAt(2, Character.toUpperCase(sMethod.charAt(2))); // upper case the first
      character of the action
00048             if (Text.compare(sMethod, "doPost") == 0)
00049             {
00050                 oRep.sendError(401);
00051                 return;
00052             }
00053 
00054             for (Method oMethod : getClass().getDeclaredMethods())
00055             {
00056                 if (Text.compare(sMethod, oMethod.getName()) == 0)
00057                 {
00058                     try (PrintWriter oOut = oRep.getWriter())
00059                     {
00060                         oMethod.invoke(this, oSess, oReq, oOut);
00061                         return;
00062                     }
00063                 }
00064             }
00065             oRep.sendError(401);
00066         }
00067         catch (Exception oEx)
00068         {
00069             oRep.sendError(409);
00070         }
00071     }
00072 
00073 
00074     public static void update(String[] sObj, int nCols, CsvReader oIn)
00075     {
00076         int nLimit = Math.min(nCols, sObj.length);
00077         for (int i = 0; i < nLimit; i++)
00078             sObj[i] = oIn.parseString(i);
00079     }
00080 
00081 
00082     public static void writeJson(PrintWriter oOut, String sId, String sData)
00083     {
00084         oOut.write(String.format("\"%s\":%s", sId, sData));
00085     }
00086 }
```

8.22 src/cc/ws/ReplayMgr.java File Reference

Classes

- class cc.ws.ReplayMgr
- class cc.ws.ReplayMgr.Storm

Packages

- package cc.ws

8.23 ReplayMgr.java

Go to the documentation of this file.

```
00001 package cc.ws;
00002 
00003 import java.io.FileInputStream;
00004 import java.io.IOException;
00005 import java.text.SimpleDateFormat;
00006 import java.util.ArrayList;
00007 import javax.servlet.http.HttpServlet;
00008 import javax.servlet.http.HttpServletRequest;
00009 import javax.servlet.http.HttpServletResponse;
00010 import javax.json.Json;
```

```
00011 import javax.json.stream.JsonGenerator;
00012 import cc.util.CsvReader;
00013
00014
00015 public class ReplayMgr extends HttpServlet implements Runnable
00016 {
00017     protected String m_sStormFile;
00018     private final ArrayList<Storm> m_oStorms = new ArrayList();
00019
00020
00021     public ReplayMgr()
00022     {
00023     }
00024
00025
00026     @Override
00027     public void init()
00028     {
00029         String sStormFile = getServletConfig().getInitParameter("stormfile");
00030         if (sStormFile != null && sStormFile.length() > 0)
00031         {
00032             m_sStormFile = sStormFile;
00033             new Thread(this).start();
00034         }
00035     }
00036
00037
00038     @Override
00039     public void run()
00040     {
00041         int nCells = Integer.MIN_VALUE;
00042         try (CsvReader oIn = new CsvReader(new FileInputStream(m_sStormFile)))
00043         {
00044             SimpleDateFormat oFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm");
00045             StringBuilder sBuf = new StringBuilder();
00046
00047             oIn.readLine(); // skip header: start end avg max min
00048             while (oIn.readLine() > 0)
00049             {
00050                 oIn.parseString(sBuf, 0); // get start date as string
00051                 String sStart = sBuf.toString();
00052
00053                 oIn.parseString(sBuf, 1); // get end date as string
00054                 String sEnd = sBuf.toString();
00055
00056                 double dDur = oFormat.parse(sEnd).getTime() - oFormat.parse(sStart).getTime();
00057
00058                 Storm oStorm = new Storm();
00059                 oStorm.m_sStart = sStart;
00060                 oStorm.m_sEnd = sEnd;
00061                 oStorm.m_sHours = String.format("%03.1f", dDur / 3600000.0);
00062                 oStorm.m_nAvg = (int)Math.round(oIn.parseDouble(2));
00063                 oStorm.m_nMax = oIn.parseInt(3);
00064                 oStorm.m_nMin = oIn.parseInt(4);
00065                 m_oStorms.add(oStorm);
00066
00067                 if (oStorm.m_nMax > nCells)
00068                     nCells = oStorm.m_nMax; // find greatest cell count
00069             }
00070         }
00071         catch (Exception oEx)
00072         {
00073             oEx.printStackTrace();
00074         }
00075
00076         if (nCells <= 0)
00077             return;
00078
00079         for (Storm oStorm : m_oStorms)
00080         {
00081             oStorm.m_nAvg = 100 * oStorm.m_nAvg / nCells;
00082             oStorm.m_nMin = 100 * oStorm.m_nMin / nCells;
00083             oStorm.m_nMax = 100 * oStorm.m_nMax / nCells;
00084         }
00085     }
00086
00087
00088     @Override
00089     public void doPost(HttpServletRequest oReq, HttpServletResponse oRep)
00090         throws IOException
00091     {
00092         Session oSess = SessMgr.getSession(oReq);
00093         if (oSess == null)
00094         {
00095             oRep.sendError(401);
00096             return;
00097         }
```

```
00098
00099         try (JsonGenerator oJson = Json.createGenerator(oRep.getOutputStream()))
00100         {
00101             oJson.writeStartArray(); // start outer JSON array
00102             for (Storm oStorm : m_oStorms)
00103             {
00104                 oJson.writeStartArray(); // start record
00105                 oJson.write(oStorm.m_sStart).write(oStorm.m_sEnd).write(oStorm.m_sHours).
00106                     write(oStorm.m_nAvg).write(oStorm.m_nMax).write("");
00107                 oJson.writeEnd(); // end record
00108             }
00109             oJson.writeEnd(); // end outer JSON array
00110         }
00111     }
00112
00113
00114     private class Storm
00115     {
00116         public String m_sStart;
00117         public String m_sEnd;
00118         public String m_sHours;
00119         public int m_nMin;
00120         public int m_nMax;
00121         public int m_nAvg;
00122
00123
00124         Storm()
00125         {
00126         }
00127     }
00128
00129
00130 //  public static void main(String[] sArgs)
00131 //  {
00132 //      ReplayMgr oMgr = new ReplayMgr();
00133 //      oMgr.m_sStormFile = "C:/Users/bryan.krueger/2018storms.csv";
00134 //      oMgr.run();
00135 //      System.out.println(oMgr.m_oStorms.size());
00136 //  }
00137 }
```

8.24 src/cc/ws/RopMgr.java File Reference

Classes

- class cc.ws.RopMgr

Packages

- package cc.ws

8.25 RopMgr.java

Go to the documentation of this file.

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005  */
00006 package cc.ws;
00007
00008 import cc.util.CsvReader;
00009 import cc.util.Text;
00010 import java.io.BufferedWriter;
00011 import java.io.FileInputStream;
00012 import java.io.FileWriter;
00013 import java.io.IOException;
00014 import java.io.PrintWriter;
00015 import java.util.ArrayList;
00016 import java.util.Collections;
```

```
00017 import javax.servlet.http.HttpServletRequest;
00018
00023 public class RopMgr extends Handler implements Runnable
00024 {
00025     protected String m_sRopFile;
00026     protected ArrayList<String[]> m_oRops = new ArrayList();
00027
00028
00029     @Override
00030     public void init()
00031     {
00032         String sRopFile = getServletConfig().getInitParameter("ropfile");
00033         if (sRopFile != null && sRopFile.length() > 0)
00034         {
00035             m_sRopFile = sRopFile;
00036             new Thread(this).start();
00037         }
00038     }
00039
00040
00041     @Override
00042     public void run()
00043     {
00044         try (CsvReader oIn = new CsvReader(new FileInputStream(m_sRopFile), '\t'))
00045         {
00046             int nCols;
00047             String[] sSearch = new String[1];
00048             synchronized(m_oRops)
00049             {
00050                 while ((nCols = oIn.readLine()) > 0)
00051                 {
00052                     sSearch[0] = oIn.parseString(0);
00053                     int nIndex = Collections.binarySearch(m_oRops, sSearch, STR_ARR_COMP); // search
      for id in list
00054                     if (nIndex < 0)
00055                     {
00056                         nIndex = ~nIndex;
00057                         m_oRops.add(nIndex, new String[nCols]); // right now we have
      uuid,user,timestamp,status,JSON
00058                     }
00059                     update(m_oRops.get(nIndex), nCols, oIn); // replace with most recent update
00060                 }
00061             }
00062         }
00063         catch (Exception oEx)
00064         {
00065         }
00066     }
00067
00068
00069     protected void doNull(Session oSess, HttpServletRequest oReq, PrintWriter oOut)
00070     {
00071         oOut.write("{");
00072         synchronized (m_oRops)
00073         {
00074             int nSize = m_oRops.size();
00075             if (nSize > 0)
00076             {
00077                 String[] sRop = m_oRops.get(0);
00078                 writeJson(oOut, sRop[0], sRop[sRop.length - 1]);
00079                 for (int i = 1; i < nSize; i++)
00080                 {
00081                     sRop = m_oRops.get(i);
00082                     oOut.write(",");
00083                     writeJson(oOut, sRop[0], sRop[sRop.length - 1]); // {"uuid" : JSON, "uuid" : JSON,
      ...}
00084                 }
00085             }
00086         }
00087         oOut.write("}");
00088     }
00089
00090
00091     protected void doSave(Session oSess, HttpServletRequest oReq, PrintWriter oOut)
00092        throws IOException
00093     {
00094         String[] sRop;
00095         synchronized (m_oRops)
00096         {
00097             String[] sSearch = new String[1];
00098             sSearch[0] = oReq.getParameter("id");
00099             if (sSearch[0] == null || sSearch[0].length() == 0) // no id so create a new one
00100                 sSearch[0] = Text.getUUID();
00101
00102             StringBuilder sBuffer = new StringBuilder(oReq.getParameter("data"));
00103             if (sBuffer.indexOf("\"status\"") < 0) // add default status of unused to json
00104             {
```

```
00105                 Text.removeWhitespace(sBuffer);
00106                 sBuffer.insert(sBuffer.length() - 2, ",\"status\":\"U\"");
00107             }
00108 
00109             int nIndex = Collections.binarySearch(m_oRops, sSearch, STR_ARR_COMP);
00110             if (nIndex < 0)
00111             {
00112                 nIndex = ~nIndex;
00113                 m_oRops.add(nIndex, new String[]{sSearch[0], oSess.m_oUser.m_sUser, null,
      sBuffer.toString()}); // create new array
00114             }
00115             sRop = m_oRops.get(nIndex);
00116             synchronized (ISO8601Sdf)
00117             {
00118                 sRop[2] = ISO8601Sdf.format(System.currentTimeMillis());
00119             }
00120 
00121             if (sRop[3].contains("\"status\":\"D\"")) // remove from list if the status is deleted
00122                 m_oRops.remove(nIndex);
00123 
00124             try (BufferedWriter oFileOut = new BufferedWriter(new FileWriter(m_sRopFile, true)))
00125             {
00126                 oFileOut.write(sRop[0]);
00127                 for (int i = 1; i < sRop.length; i++)
00128                 {
00129                     oFileOut.write("\t");
00130                     oFileOut.write(sRop[i]);
00131                 }
00132                 oFileOut.write("\n");
00133             }
00134         }
00135         oOut.write("{");
00136         writeJson(oOut, sRop[0], sRop[sRop.length - 1]); // id:data
00137         oOut.write("}");
00138     }
00139 }
```

8.26 src/cc/ws/Session.java File Reference

Classes

- class cc.ws.Session

Packages

- package cc.ws

8.27 Session.java

Go to the documentation of this file.

```
00001 package cc.ws;
00002 
00003 import java.util.Comparator;
00004 
00005 
00006 public class Session implements Comparable<String>, Comparator<Session>
00007 {
00008     protected long m_lUpdate;
00009     protected String m_sToken;
00010     protected User m_oUser;
00011 
00012 
00013     protected Session()
00014     {
00015     }
00016 
00017 
00018     Session(String sKey)
00019     {
00020         m_sToken = sKey;
```

```
00021     }
00022
00023
00024     @Override
00025     public int compareTo(String sKey)
00026     {
00027         return m_sToken.compareTo(sKey);
00028     }
00029
00030
00031     @Override
00032     public int compare(Session oLhs, Session oRhs)
00033     {
00034         return oLhs.m_sToken.compareTo(oRhs.m_sToken);
00035     }
00036 }
```

8.28 src/cc/ws/SessMgr.java File Reference

Classes

- class cc.ws.SessMgr

Packages

- package cc.ws

8.29 SessMgr.java

Go to the documentation of this file.

```
00001 package cc.ws;
00002
00003 import java.security.SecureRandom;
00004 import java.util.ArrayList;
00005 import java.util.Collections;
00006 import javax.servlet.ServletConfig;
00007 import javax.servlet.http.HttpServlet;
00008 import javax.servlet.http.HttpServletRequest;
00009 import cc.util.Text;
00010
00011
00012 public class SessMgr extends HttpServlet
00013 {
00014     static long TIMEOUT = 1800000; // default 30-minute sessions
00015     static final ArrayList<Session> SESSIONS = new ArrayList();
00016
00017
00018     public SessMgr()
00019     {
00020     }
00021
00022
00023     @Override
00024     public void init()
00025     {
00026         ServletConfig oConf = getServletConfig();
00027         String sExp = oConf.getInitParameter("timeout");
00028         if (sExp != null)
00029             TIMEOUT = Text.parseInt(sExp);
00030     }
00031
00032
00033     static Session getSession(HttpServletRequest oReq)
00034     {
00035         return getSession(oReq, false);
00036     }
00037
00038
00039     static Session getSession(HttpServletRequest oReq, boolean bCreate)
00040     {
```

```
00041         String sToken = oReq.getParameter("token");
00042         if (sToken == null)
00043             sToken = "";
00044 
00045         Session oSess = null;
00046         synchronized(SESSIONS)
00047         {
00048             int nIndex = Collections.binarySearch(SESSIONS, sToken);
00049             if (nIndex >= 0)
00050             {
00051                 oSess = SESSIONS.get(nIndex);
00052                 if (oSess.m_lUpdate < System.currentTimeMillis())
00053                 {
00054                     SESSIONS.remove(nIndex);
00055                     return null;
00056                 }
00057             }
00058             else if (bCreate)
00059             {
00060                 byte[] yBytes = new byte[16];
00061                 try
00062                 {
00063                     SecureRandom oRng = SecureRandom.getInstance("SHA1PRNG");
00064                     do
00065                     {
00066                         oRng.nextBytes(yBytes); // ensure no duplicates
00067                         oSess = new Session(Text.toHexString(yBytes));
00068                         nIndex = Collections.binarySearch(SESSIONS, oSess, oSess);
00069                     }
00070                     while (nIndex >= 0);
00071                     SESSIONS.add(~nIndex, oSess); // save new session
00072                 }
00073                 catch (Exception oEx)
00074                 {
00075                 }
00076             }
00077         }
00078 
00079         if (oSess != null)
00080             oSess.m_lUpdate = System.currentTimeMillis() + TIMEOUT;
00081 
00082         return oSess;
00083     }
00084 
00085 
00086     static void removeSession(Session oSess)
00087     {
00088         if (oSess == null)
00089             return;
00090 
00091         synchronized(SESSIONS)
00092         {
00093             int nIndex = Collections.binarySearch(SESSIONS, oSess, oSess);
00094             if (nIndex >= 0)
00095                 SESSIONS.remove(nIndex);
00096         }
00097     }
00098 }
```

8.30 src/cc/ws/User.java File Reference

Classes

- class cc.ws.User

Packages

- package cc.ws

8.31 User.java

Go to the documentation of this file.

```
00001 package cc.ws;
00002
00003 import cc.util.CsvReader;
00004 import cc.util.Text;
00005 import java.util.Comparator;
00006
00007
00008 public class User implements Comparable<String>, Comparator<User>
00009 {
00010     byte[] m_ySalt;
00011     String m_sUser;
00012     String m_sPass;
00013     String m_sGroup;
00014
00015
00016     User()
00017     {
00018     }
00019
00020
00021     User(CsvReader oCsv)
00022     {
00023         StringBuilder sCol = new StringBuilder();
00024         oCsv.parseString(sCol, 0);
00025         m_sUser = sCol.toString(); // save username
00026
00027         oCsv.parseString(sCol, 1);
00028         m_ySalt = Text.fromHexString(sCol);
00029
00030         oCsv.parseString(sCol, 2);
00031         m_sPass = sCol.toString(); // keep password as hexadecimal string
00032
00033         oCsv.parseString(sCol, 3);
00034         m_sGroup = sCol.toString().intern(); // very few group patterns
00035     }
00036
00037
00038     @Override
00039     public int compareTo(String sUser)
00040     {
00041         return m_sUser.compareTo(sUser);
00042     }
00043
00044
00045     @Override
00046     public int compare(User oLhs, User oRhs)
00047     {
00048         return oLhs.m_sUser.compareTo(oRhs.m_sUser);
00049     }
00050 }
```

8.32 src/cc/ws/UserMgr.java File Reference

Classes

- class cc.ws.UserMgr

Packages

- package cc.ws

8.33 UserMgr.java

Go to the documentation of this file.

```
00001 package cc.ws;
00002
00003 import java.io.FileInputStream;
00004 import java.security.MessageDigest;
00005 import java.util.ArrayList;
00006 import java.util.Collections;
00007 import javax.servlet.ServletConfig;
00008 import javax.servlet.ServletOutputStream;
00009 import javax.servlet.http.HttpServlet;
00010 import javax.servlet.http.HttpServletRequest;
00011 import javax.servlet.http.HttpServletResponse;
00012 import cc.util.CsvReader;
00013 import cc.util.Text;
00014
00015
00016 public class UserMgr extends HttpServlet
00017 {
00018     protected static final Object LOCK = new Object();
00019     protected static MessageDigest DIGEST;
00020     protected ArrayList<User> m_oCreds = new ArrayList();
00021
00022     static
00023     {
00024         try
00025         {
00026             DIGEST = MessageDigest.getInstance("SHA-256");
00027         }
00028         catch (Exception oEx)
00029         {
00030         }
00031     }
00032
00033     public UserMgr()
00034     {
00035     }
00036
00037
00038     @Override
00039     public void init()
00040     {
00041         ServletConfig oConf = getServletConfig();
00042         try (CsvReader oCsv = new CsvReader(new FileInputStream(oConf.getInitParameter("pwdfile"))))
00043         {
00044             while (oCsv.readLine() > 0)
00045                 m_oCreds.add(new User(oCsv));
00046         }
00047         catch (Exception oEx)
00048         {
00049         }
00050         Collections.sort(m_oCreds, new User());
00051     }
00052
00053
00054     @Override
00055     public void doPost(HttpServletRequest oReq, HttpServletResponse oRep)
00056     {
00057         StringBuilder sBuf = new StringBuilder("{\n");
00058         String sPath = oReq.getPathInfo();
00059         if (sPath.contains("login"))
00060         {
00061             String sUname = oReq.getParameter("uname");
00062             String sPword = oReq.getParameter("pword");
00063             if (sUname != null && sUname.length() > 0 && sPword != null && sPword.length() > 0)
00064             {
00065                 int nIndex = Collections.binarySearch(m_oCreds, sUname);
00066                 if (nIndex >= 0)
00067                 {
00068                     User oUser = m_oCreds.get(nIndex);
00069                     StringBuilder sSecPass = new StringBuilder();
00070                     getSecurePassword(sPword, oUser.m_ySalt, sSecPass);
00071                     if (Text.compare(oUser.m_sPass, sSecPass) == 0)
00072                     {
00073                         Session oSess = SessMgr.getSession(oReq, true);
00074                         oSess.m_oUser = oUser; // save credentials in session
00075                         sBuf.append("\t\"token\": \"").append(oSess.m_sToken).append("\"\n");
00076                     }
00077                 }
00078             }
00079         }
00080         else
00081         {
00082             Session oSess = SessMgr.getSession(oReq);
```

```
00083             if (sPath.contains("check"))
00084             {
00085                 if (oSess != null)
00086                     sBuf.append("\t\"token\": \"").append(oSess.m_sToken).append("\"\n");
00087             }
00088             else if (sPath.contains("logout"))
00089             {
00090                     SessMgr.removeSession(oSess);
00091             }
00092             else if (sPath.contains("update"))
00093             {
00094             }
00095         }
00096         sBuf.append("}\n");
00097 
00098         try (ServletOutputStream oOut = oRep.getOutputStream())
00099         {
00100             for (int nIndex = 0; nIndex < sBuf.length(); nIndex++)
00101                 oOut.print(sBuf.charAt(nIndex));
00102         }
00103         catch (Exception oEx)
00104         {
00105         }
00106     }
00107 
00108 
00109     static void getSecurePassword(String sPass, byte[] ySalt, StringBuilder sBuf)
00110     {
00111         synchronized(LOCK)
00112         {
00113             DIGEST.reset();
00114             DIGEST.update(ySalt);
00115             Text.toHexString(DIGEST.digest(sPass.getBytes()), sBuf);
00116         }
00117     }
00118 
00119 
00120     public static void main(String[] sArgs)
00121     {
00122         String sUser = sArgs[0];
00123         String sPass = sArgs[1];
00124 
00125         byte[] ySalt = new byte[32]; // use 256-bit algorithm
00126         try
00127         {
00128             java.security.SecureRandom.getInstance("SHA1PRNG").nextBytes(ySalt);
00129             StringBuilder sBuf = new StringBuilder();
00130             UserMgr.getSecurePassword(sPass, ySalt, sBuf);
00131 
00132             System.out.print(sUser);
00133             System.out.print(",");
00134             System.out.print(Text.toHexString(ySalt));
00135             System.out.print(",");
00136             System.out.print(sBuf.toString());
00137             System.out.print(",");
00138             System.out.println("abcdefghijklmnopqrstWvwxyz");
00139         }
00140         catch (Exception oEx)
00141         {
00142         }
00143     }
00144 }
```

Index