

Jennifer 1.0 TOMA Manual

JenniferSoft, Inc.

3.2

Copyright © 2007 JenniferSoft, Inc.

Table of Contents

1. Getting Started	1
2. Installation and Execution	2
2.1. Environment Setting and Parser Registration	2
3. Analysis of the Apache Access Logs	4
3.1. Log Import	4
3.2. Distribution Graph	6
3.3. Analysis Graph	7
3.4. Filtering	9
4. Parser Program Guide	11
4.1. Interface	11
4.1.1. public void init();	12
4.1.2. public void end();	12
4.1.3. public XViewItem parse(String str);	12
4.1.4. public XViewItem parse(byte[] bytes);	13
4.1.5. public String[] getValueNames();	13
4.1.6. public byte[] getValueTypes();	13
4.1.7. public String[] getKeywordNames();	13
4.1.8. XViewItem->public long time;	13
4.1.9. XViewItem->public String name;	13
4.1.10. XViewItem->public double[] values;	13
4.1.11. XViewItem->public String[] keywords;	14
4.1.12. XViewItem->public String contents;	14
4.2. The Use of Parser in Toma	14
4.3. An example of the Apache access log parser	15
5. Importing Jennifer 3.x X-View Data	18
5.1. Profile Data Download	18
5.2. Analysis of Appls.dat and sql.dat, methods.dat	18
5.2.1. Execution Process	18
5.3. Importing in Toma	18
6. Importing Jennifer 2.5.x X-View Data	20
6.1. Profile Data Download	20
6.2. Analysis of Appls.dat and sql.dat	20
6.2.1. Execution Process	20
6.3. Importing in Toma	20
7. Using the JarUtil	23
7.1. Search Class/Method	23
7.2. Search Class/Method User	24
7.3. Search Static Collection	25
7.4. Search Native Method	26
7.5. Class Diagram	26
7.6. Byte Code	27

1. Getting Started

Toma is a sub-module of Jennifer. Toma provides useful functions for operating Jennifer and analyzing various kinds of problems.

2. Installation and Execution

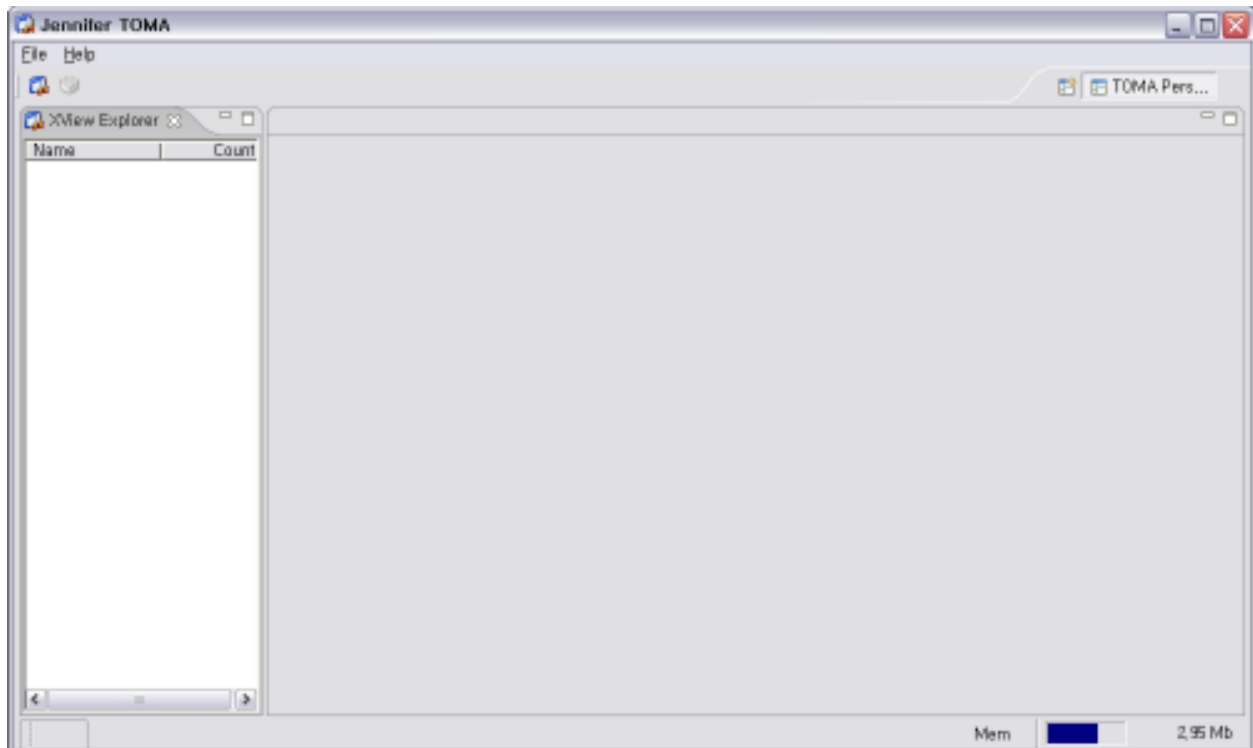
Jennifer's Toma is distributed in a zip file format. Installation is completed by extracting the zip file to your desired directory.



Important

You need to install Java 1.4 or newer to run Toma.

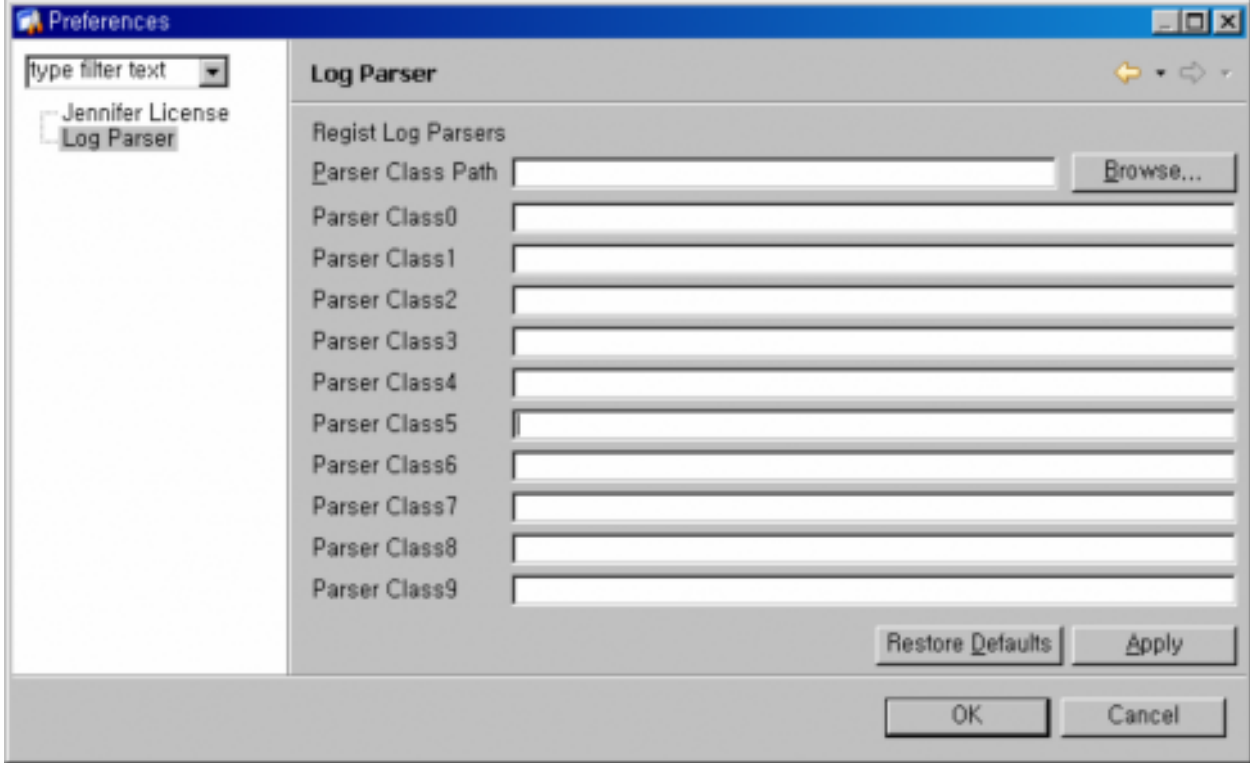
If you execute TOMA_HOME/toma.exe, you can see the initial page as follows:



Initial Page

2.1 Environment Setting and Parser Registration

Select the "Help/Preference" menu to see the configuration screen. Select "Log Parser" and you will see the parser registration screen. You can register the newly-created parser in here.



Log Parser

Register the new parser in the “Parser Class Path” field. Note that only the jar file format is allowed to register. (For example, use the sample in the TOMA_HOME/addin/ directory.)

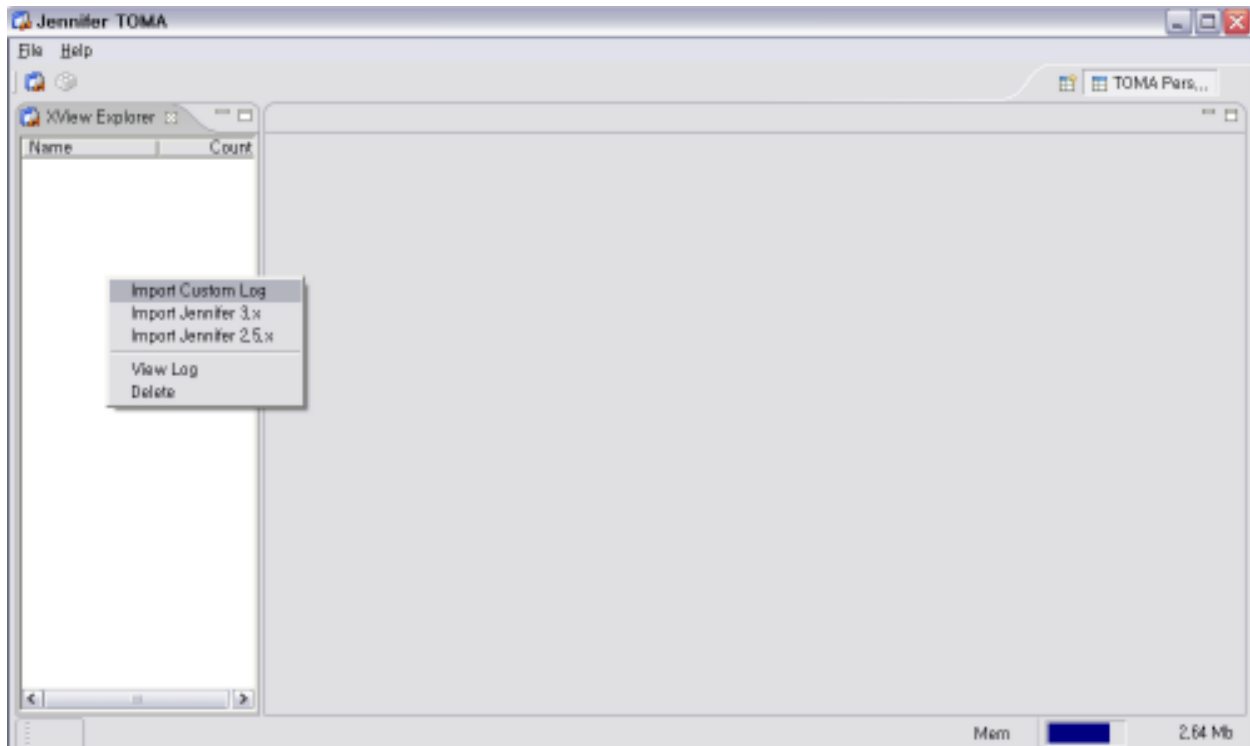
3. Analysis of the Apache Access Logs

In this section, we describe the method for analyzing the Apache access logs by importing.

3.1 Log Import

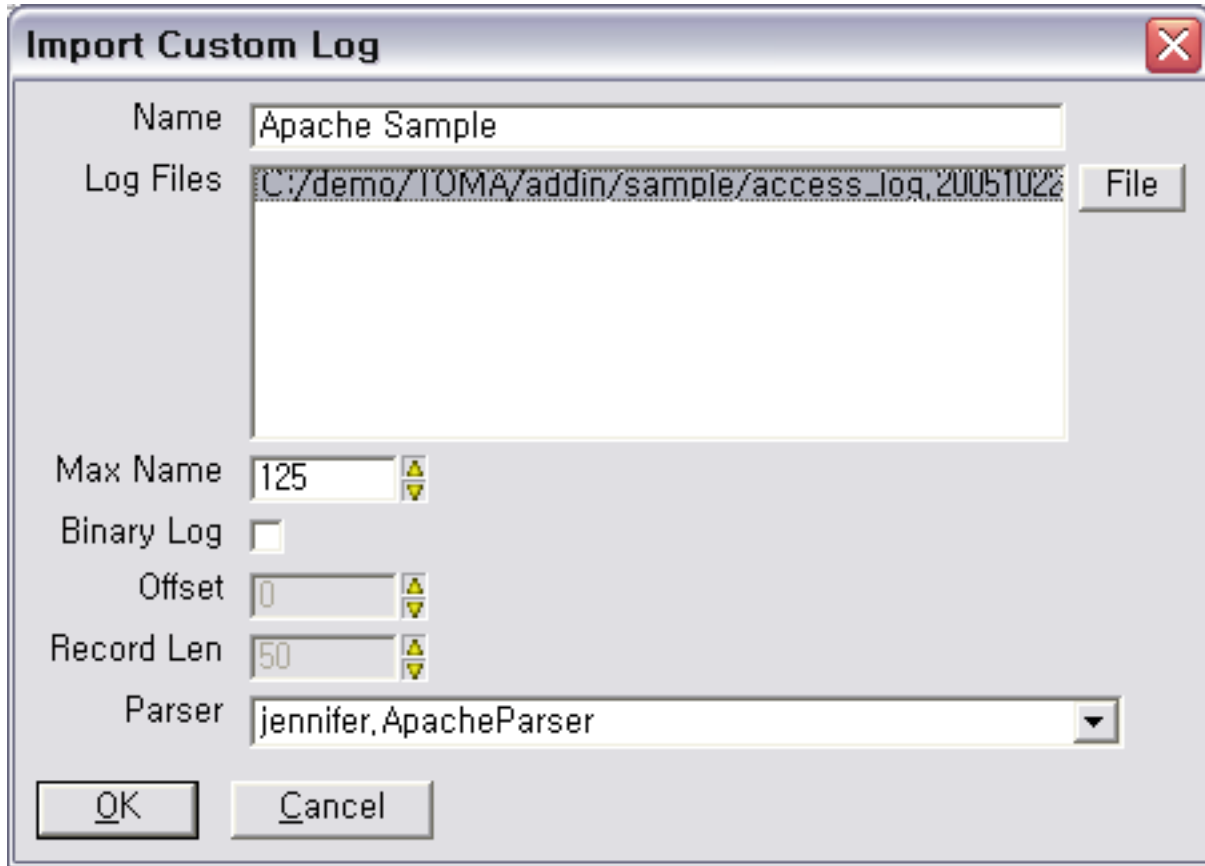
To analyze an arbitrary log by X-View, you need to execute import first. Click the right mouse button in the XViewExplorer window and you will see the following four menus:

- Import Custom Log: Import through the parser registered in “Preference.”
- Import Jennifer 3.x: Import the X-View data of Jennifer 3.x version.
- Import Jennifer2.5.x: Import the X-View data of Jennifer 2.5.x version.
- Delete: Delete the imported log.



Import

Select “Import Custom Log” because this example will analyze the Apache log.



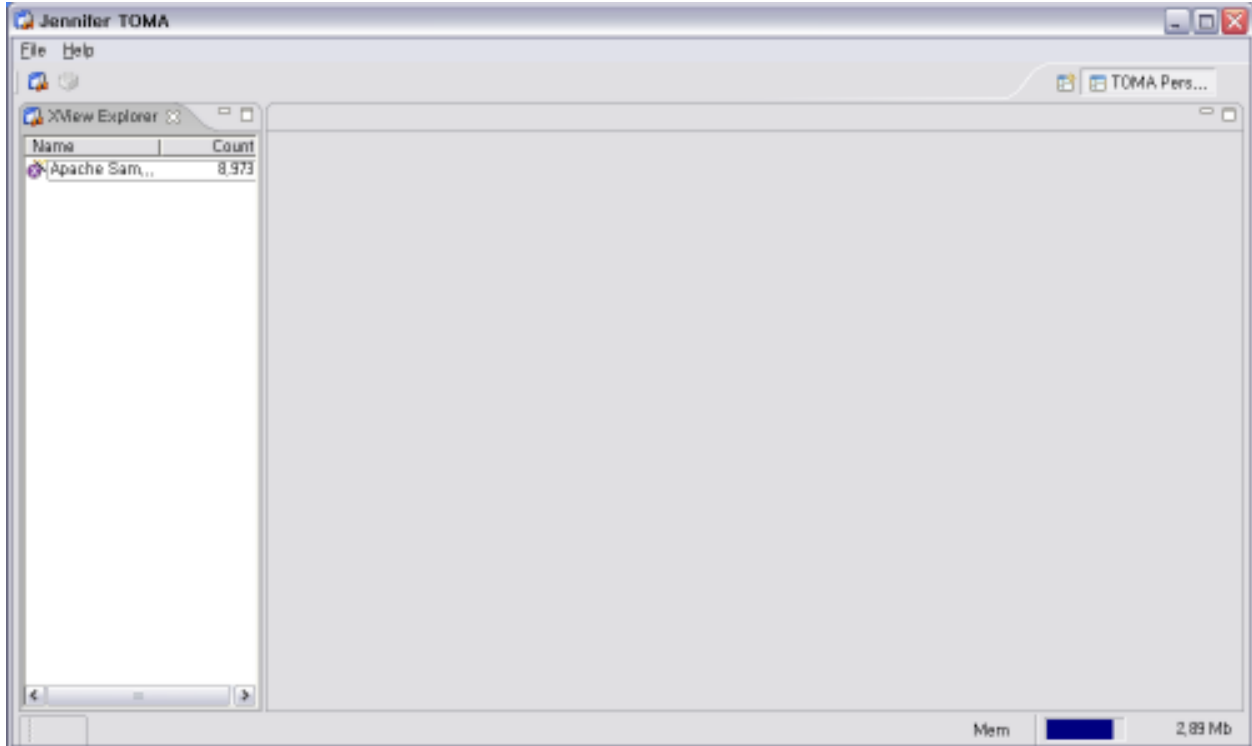
Import Properties

- Name: The name to be shown when imported
- Log File: The name of the log file for analysis (Multiple logs can be imported at a time.)
- Max Name: Max length of the log item name (The URL in the case of the access log)
- Binary Log: Checks when the imported log format is not text.
- Parser: Log parser (One from the parsers registered in "Preference")



Note

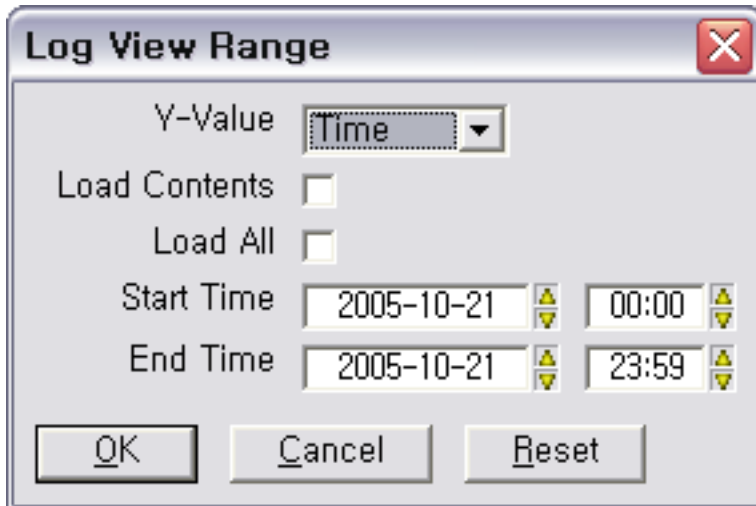
Generally, multiple Web servers run in systems. Therefore, if you want to analyze the access logs of multiple servers together, perform import at the same time. Toma merges and imports the log data by using the "INSERTION SORT" algorithm when it receives an input of multiple log files.



After Import

3.2 Distribution Graph

You can see the distribution graph of logs when you double-click the list updated in the XViewExplorer.

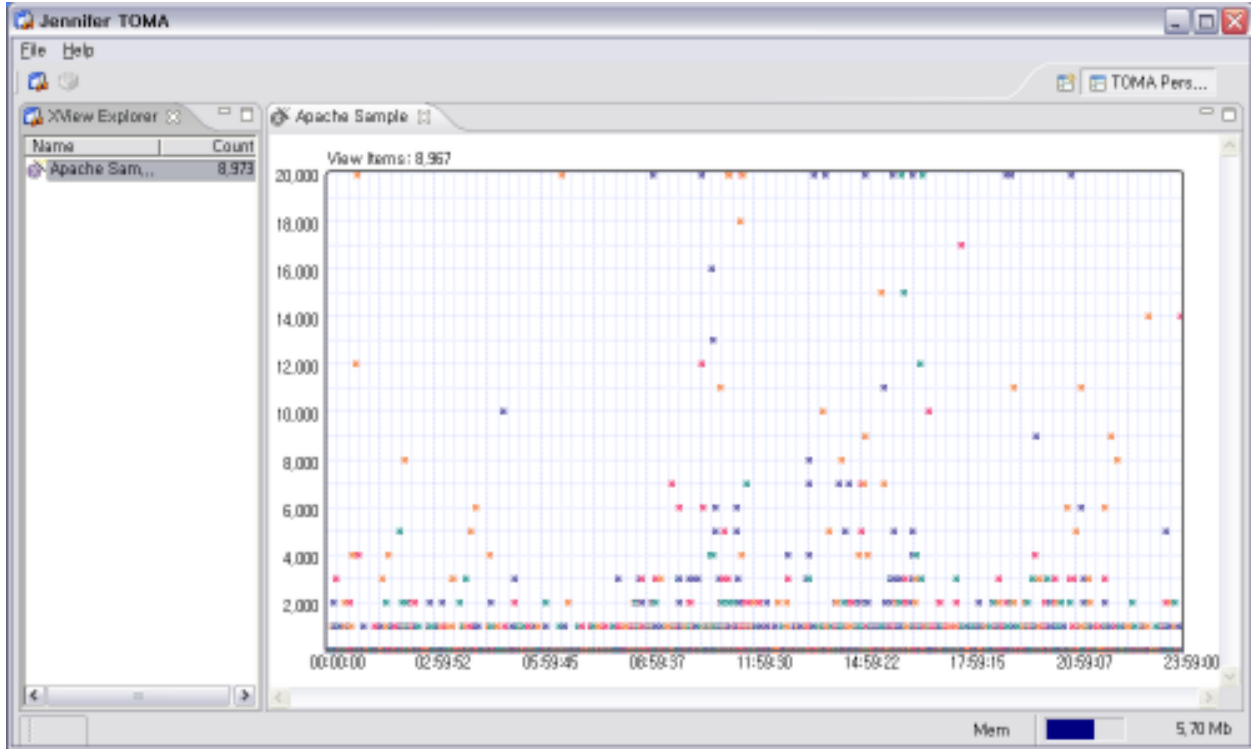


Log View

- Y-Value: The Y-axis value of the access log distribution graph. You can choose either Bytes or Time.
- Load Contents: Every imported log is separated into the head and contents parts. If you load the contents together, the filtering speed is improved even though the memory usage increases.
- Generally, "false" (deselect) is recommended.

- Load All: It means all logs are loaded. Although there is little difference in the actual loading objects, the loading speed is a little faster when you load all.

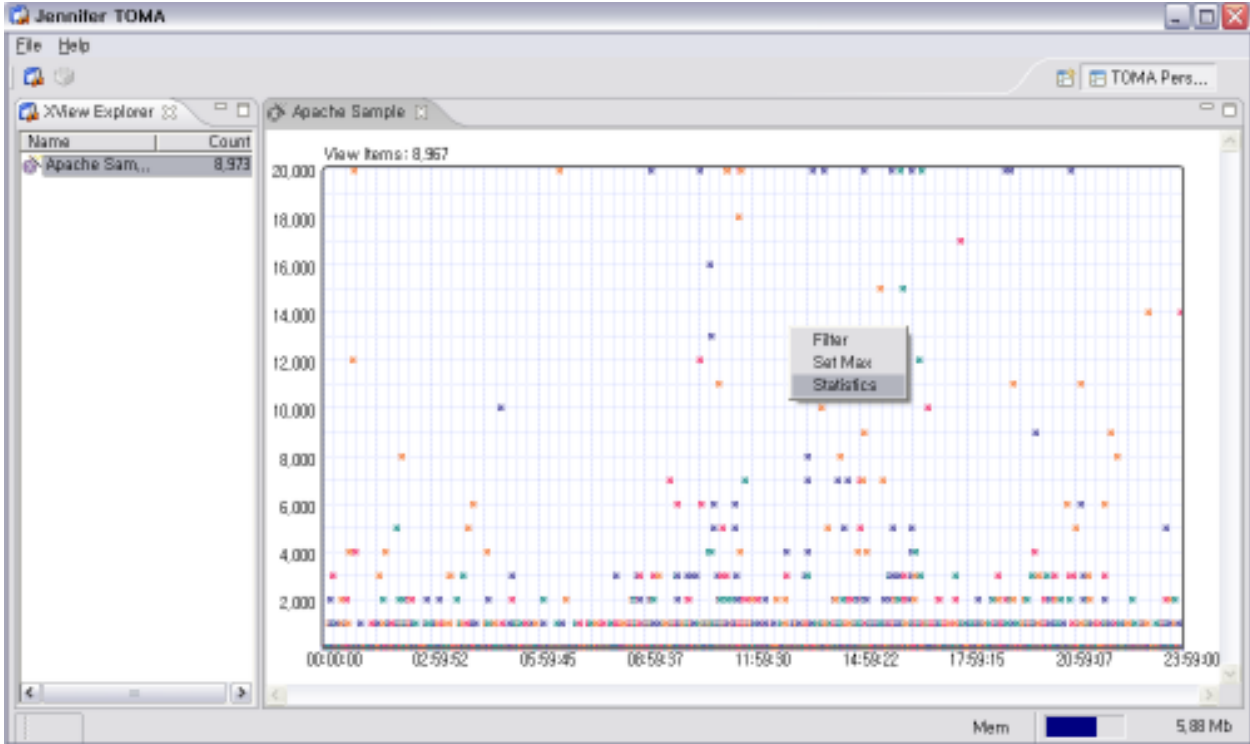
If you input the value and click “OK,” you will see the screen as follows:



X-View

3.3 Analysis Graph

You can draw various analysis graphs by using the loaded data. Toma supports three kinds of analysis graphs.



Pop-up menu

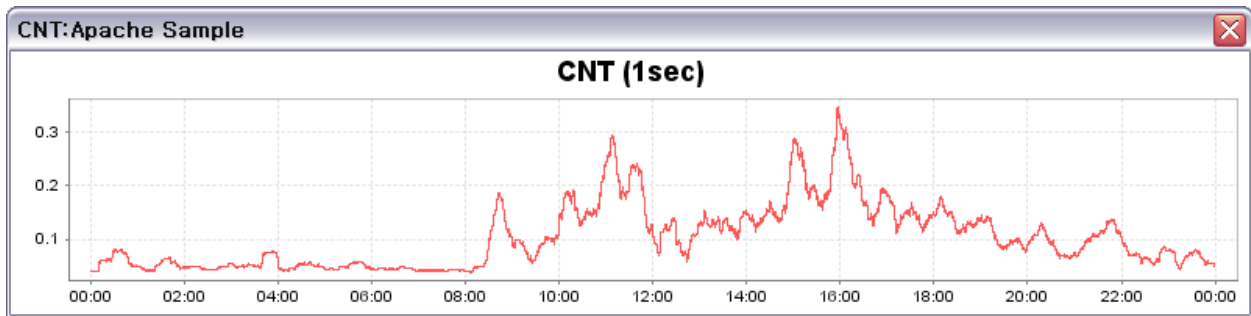
In the screen, click the right mouse button and select “Statistics.”

Input Properties

You can see the graph if you select “Graph” and click “OK” in the input screen.

- Type: The graph type (CNT, AVG, SUM)

- Calc Unit: It means the calculation unit. For example, if you set the “Calc Unit” as “1” and the “Type” as “CNT,” it means the counts per second. If you set the “Type” as “AVG,” it means the average per second. If you want to see the execution counts per hour in the graph, select “3600.”
- Normalize Unit: This option allows you to draw a graph by averaging the value calculated as the Calc Unit (sec) by the number of Normalize Unit (number).
- Normalize Cnt: Set the frequency of Normalization. As you increase the frequency, you will see a much smoother curve in the graph.
- Value: It means the value to be used on the graph. However, as “CNT” means the number, the same graph will be drawn regardless of the Value options.
- Width/Height: Set the size of the graph.



Counts per Second

3.4 Filtering

You can filter the logs loaded in the screen. If you select “Filter” by pressing the right mouse button and input appropriate values in each field, only those data meeting your conditions is shown in the window.



Note

The data is filtered by using the AND calculation of all the input values.

Filter

- Name: It is the name of each log. You can execute filtering by inputting a part of the URL in the case of the Apache log since Toma uses the indexOf method.
- Contents: Executes filtering based on the log contents. Note that the process may take a long time unless you select “Load Contents” when you load the contents.
- KEY:xxx : Executes filtering based on the specified log key value. Only those logs that accord with the key are filtered because they are compared by using the equals method.



Note

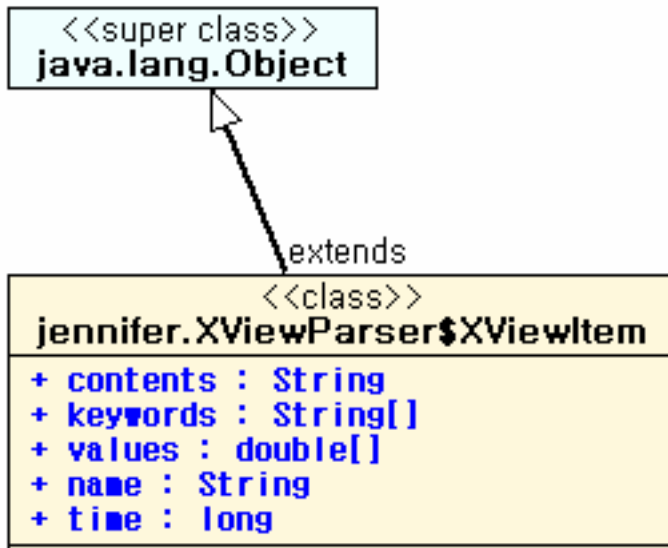
If you draw a statistical graph when the data is filtered, the graph is created only with the filtered logs.

4. Parser Program Guide

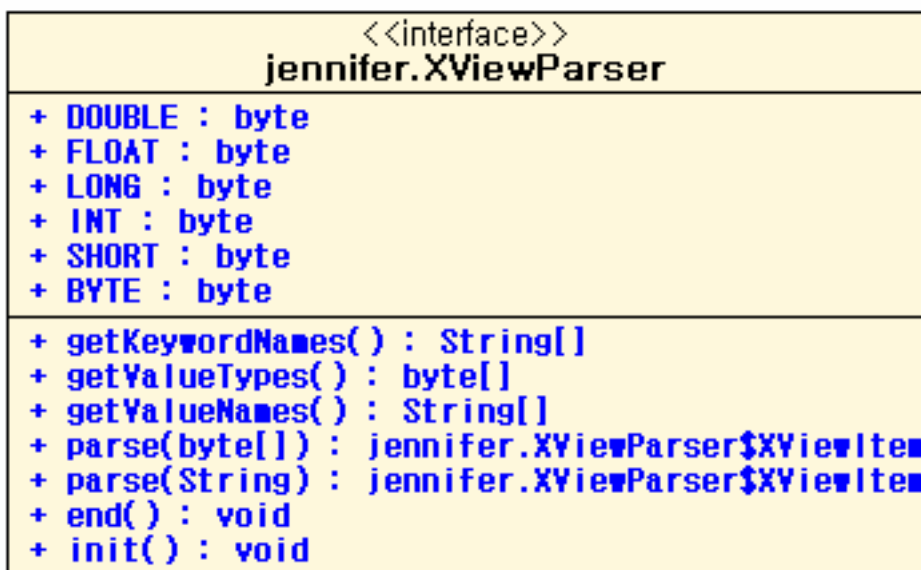
To import the user log in Toma, you need a parser. In this chapter, we will describe how to program the parser.

4.1 Interface

The X-View parser is composed of `jennifer.XViewParser` and `XViewItem`.



XViewItem



XViewParser



Important

You must make a class which implements XviewParser and use it after registering it in Preference of X-View.

```
public interface XViewParser {

    final public static byte BYTE          = 1;
    final public static byte SHORT        = 2;
    final public static byte INT           = 3;
    final public static byte LONG          = 4;
    final public static byte FLOAT         = 5;
    final public static byte DOUBLE        = 6;

    public static class XViewItem {
        public long      time;
        public String    name;
        public double[]  values;
        public String[]  keywords;
        public String    contents;
    }

    public void init();
    public void end();

    public XViewItem parse(String str);
    public XViewItem parse(byte[] bytes);

    public String[] getValueNames();
    public byte[]  getValueTypes();

    public String[] getKeywordNames();
}
```

4.1.1 public void init();

Create the initialization logic before parsing. Execute necessary initialization logic while other files are being read or before parsing.

4.1.2 public void end();

Insert memory free or file close logic after parsing is completed.

4.1.3 public XViewItem parse(String str);

Implement parsing logic for the String type log.

4.1.4 public XViewItem parse(byte[] bytes);

Implement parsing logic for the byte[] type log.

4.1.5 public String[] getValueNames();

Designate a name for each value which is included in the log.

4.1.6 public byte[] getValueTypes();

Designate a type for each Value. You can use only one type among the following list. Note that it must return the same number as getValueNames.

```
final public static byte BYTE      = 1;
final public static byte SHORT     = 2;
final public static byte INT       = 3;
final public static byte LONG      = 4;
final public static byte FLOAT     = 5;
final public static byte DOUBLE    = 6;
```

4.1.7 public String[] getKeywordNames();

It returns the keyword name required to additionally analyze specific data of the log. In case of the Apache log, use the IP or ReturnCode, etc.

4.1.8 XViewItem->public long time;

It returns the log occurrence time which is used as the value of the X-axis in the X-View screen.



Important

The imported log must be sorted by time order. If not, the data may not be shown in the screen properly. X-View does not support a sort function.

4.1.9 XViewItem->public String name;

It includes the name of the log items. In the case of the Apache access log, the URL is generally used.

4.1.10 XViewItem->public double[] values;

This is the value to be used in the log's Y-axis. In the case of the access log, Bytes and Time are generally used.



Important

This value must be the same with the return number of getValueNames().

4.1.11 XViewItem->public String[] keywords;

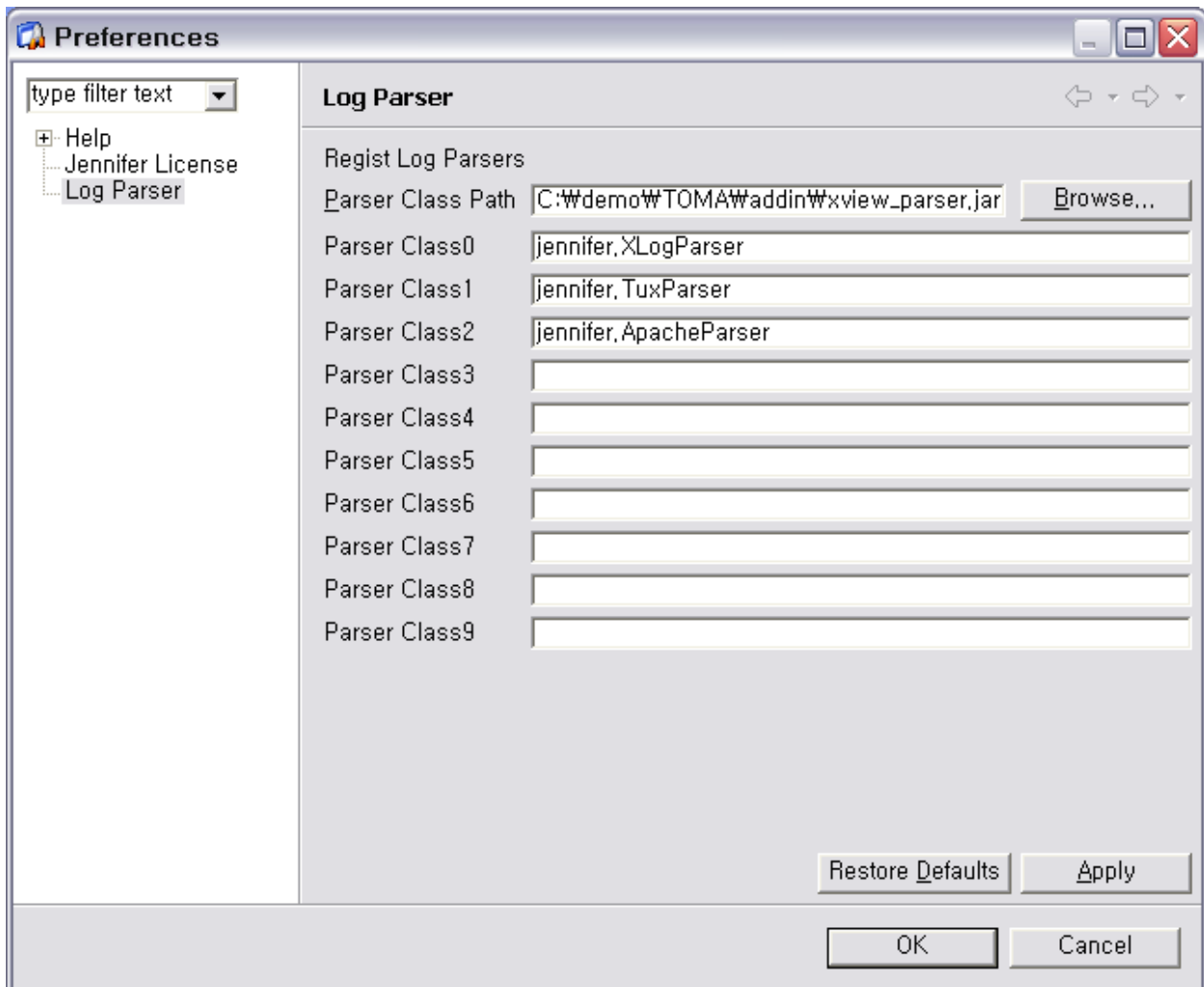
This is the value to be used for the index/keyword of the log. This value must be the same with the return number of `getKeywordNames()`.

4.1.12 XViewItem->public String contents;

This is the detailed information of the log and it may be null. In the case of null, the string which is to be sent to the parse (string) parameter is used. In the case of parse (byte[]), a proper string has to be returned.

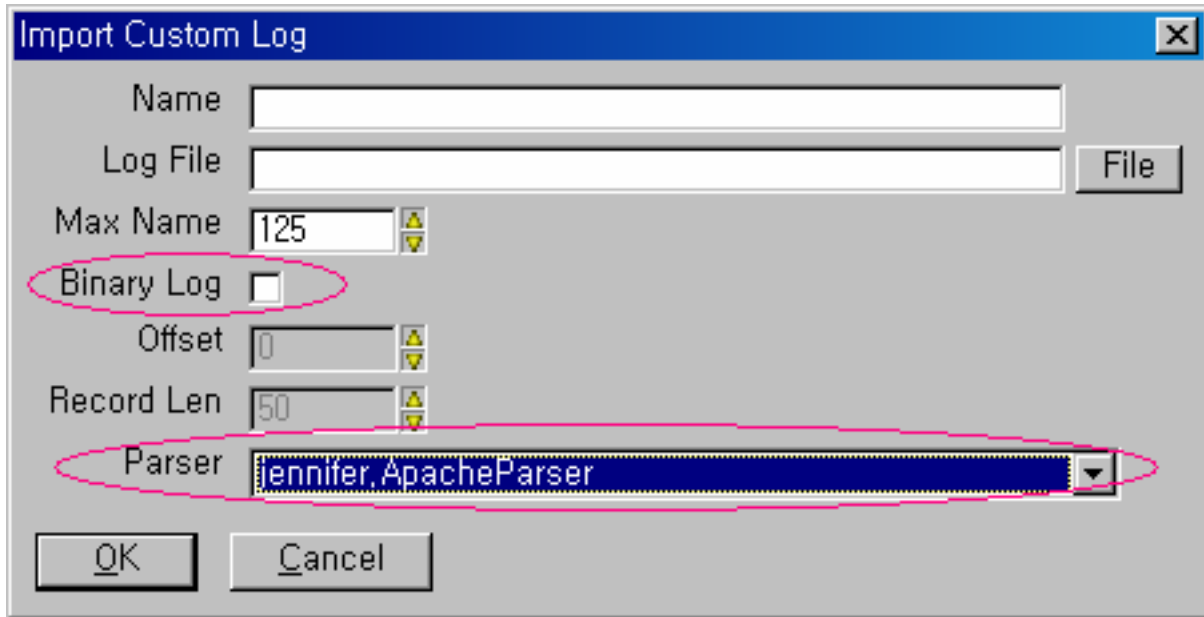
4.2 The Use of Parser in Toma

Register a parser in Preference.



Register

The following window appears when you click “Import Custom Log” in the workspace. You can select a parser registered in Preference from the “Parser” field.



Select Parser

There are two methods in the parser. If you click the “Binary Log” check box in the above window, parse (byte[]) is called, otherwise, parse (String) is called.

```
public XViewItem parse(String str);
public XViewItem parse(byte[] bytes);
```



Note

If the parse () method returns null, the data is ignored.

4.3 An example of the Apache access log parser

```
public class ApacheParser implements XViewParser {
    public XViewItem parse(String str) {
        if (str == null) //check input
            return null;
        XViewItem item = new XViewItem();
        try {
            item.name = parseURL(str); // parse URL
            item.time = parseTime(str); // parse logging time
            if (parseValue(item, str)) // parse values(byte,elapse)
                return item;
        } catch (Exception e) {
            return null;
        }
        return null;
    }
}
```

```

}

public String[] getKeywordNames() {
    return new String[] { "ip", "code" }; //additional index
}

public XViewItem parse(byte[] bline) { // not need
    return null;
}

public void init() { //not need
}

public void end() { // not need
}

public String[] getValueNames() {
    return new String[] { "Bytes", "Time" };
}

public byte[] getValueTypes() {
    return new byte[] { LONG, LONG }; //value types
}

private String parseURL(String line) throws Exception {
    int si = line.indexOf('');
    int ei = line.lastIndexOf('');
    if (si < 0 || ei < 0 || si == ei)
        throw new Exception("invalid url format");
    else
        return line.substring(si + 1, ei);
}

private long parseTime(String line) throws Exception {
    int si = line.indexOf('[');
    int ei = line.lastIndexOf(']');
    if (si < 0 || ei < 0 || si == ei)
        return 0;
    String tm = line.substring(si + 1, ei);
    SimpleDateFormat sd = new SimpleDateFormat("dd/MMM/yyyy:HH:mm:ss Z", Locale.ENGL
    return sd.parse(tm).getTime();
}

private boolean parseValue(XViewItem item, String line) throws Exception {
    int ei = line.lastIndexOf('');
    if (ei < 0 || line.length() < 20)
        return false;
    String data = line.substring(ei + 1);
    StringTokenizer st = new StringTokenizer(data, " ");
    if (st.countTokens() < 2)
        return false;
    item.keywords = new String[] { getIP(line), st.nextToken() };
    String x = st.nextToken();
    double[] v = new double[2];
    if (x.equals("-") == false)
        v[0] = Integer.parseInt(x); //parse return-bytes
}

```

```
    if (st.hasMoreTokens()) {
        x = st.nextToken();
        if (x.equals("-") == false)
            v[1] = Integer.parseInt(x) * 1000; // change time unit
    }
    item.values = v;
    return true;
}

private String getIP(String line) {
    char [] ch = new char[15];
    line.getChars(0,15,ch, 0);

    for(int i = 0 ; i < 15 ; i++){
        if(ch[i]=='-' || ch[i]==' ' || ch[i]=='_'){
            return new String(ch,0,i);
        }
    }
    return new String(ch);
}
```

5. Importing Jennifer 3.x X-View Data

Next part describes how to import Jennifer 3.x X-View profile data in Toma.

5.1 Profile Data Download

If you want to import the response time data of Jennifer 3.x in Toma, you have to copy the data to a local PC.

- XPROP.ISAM: jennifer_server/data/xprop.isam or, jennifer_server/data/yyyymdd/yyyymdd.xvd (3.03 version or newer) of the date you want to analyze or jennifer_server/data/xprof.isam
- 2007XXXX.XVW: jennifer_server/data/yyyymdd/yyyymdd.xvw of the date you want to analyze

5.2 Analysis of Appls.dat and sql.dat, methods.dat

Unlike Jennifer 2.x, the TOT.dat file of Jennifer 3.x does not contain the text information. Thus, you need to export this information by using export.jsp.

5.2.1 Execution Process

1. Copy Export.jsp to jennifer_server/webapps/ROOT
2. In the browser, execute `http://localhost:7900/export.jsp?id=a` <= Save appls.dat.
3. In the browser, execute `http://localhost:7900/export.jsp?id=s` <= Save sqls.dat.
4. In the browser, execute `http://localhost:7900/export.jsp?id=m` <= Save methods.dat.

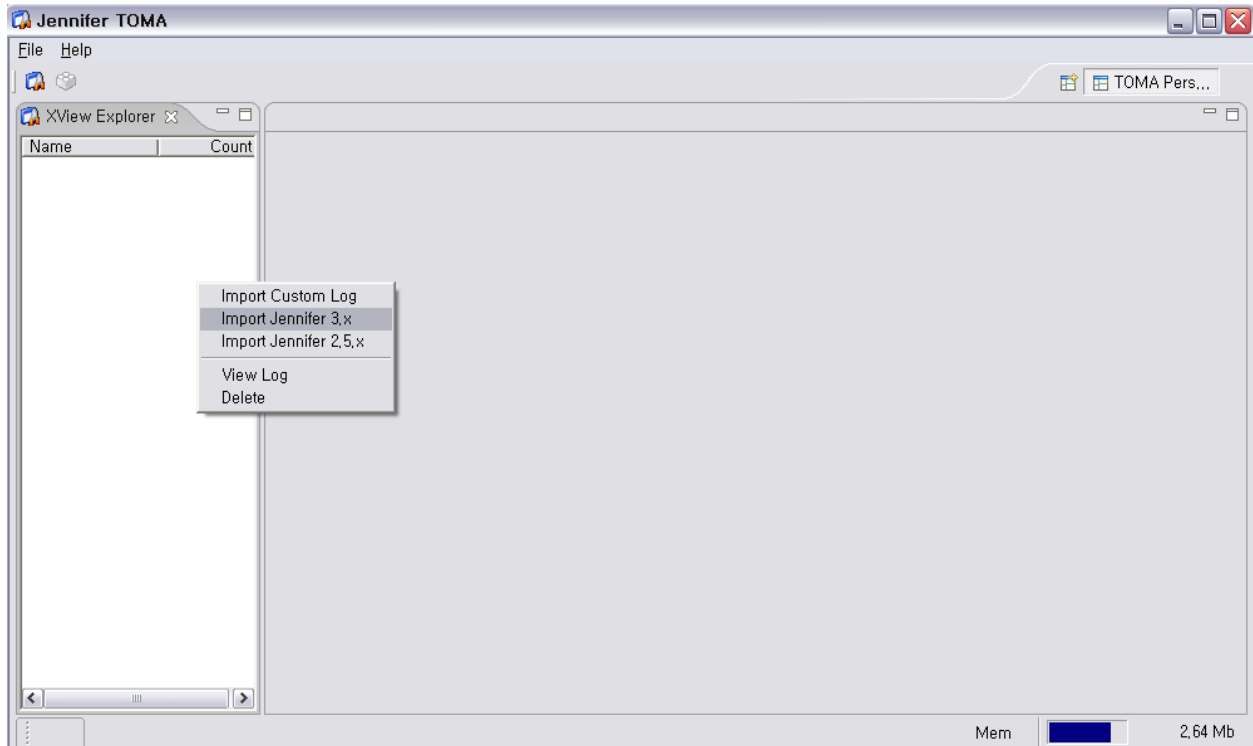


Important

If there is a security issue, delete export.jsp because it does not check log-in.

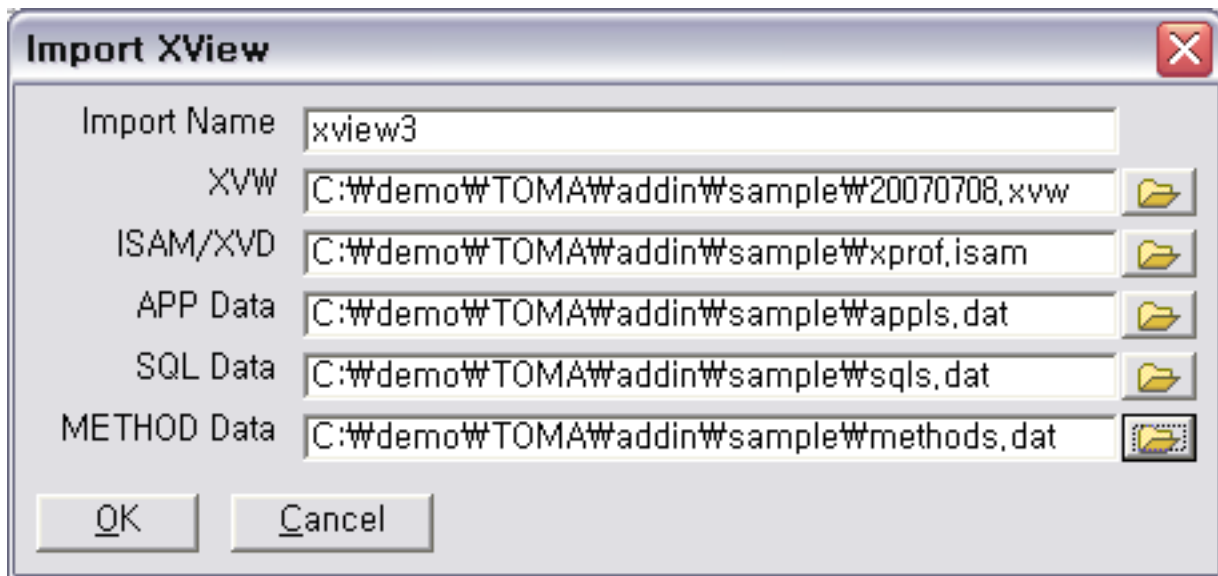
5.3 Importing in Toma

Press the right mouse button in the XviewExplorer, and select “Import Jennifer 3.x.”



Import

Enter the following information in the input window:



Files

Click OK to start the importing.

6. Importing Jennifer 2.5.x X-View Data

Next part describes how to import Jennifer 2.5.x X-View profile data in Toma.

6.1 Profile Data Download

If you want to import the response time data of Jennifer 2.5.x in Toma, you have to copy the data to a local PC.

- TOT.DAT: jennifer_server/data/yyyymmdd/TOT.dat of the date you want to analyze
- XPROP.ISAM: jennifer_server/data/xprop.isam
- 2007XXXX.XVW: jennifer_server/data/yyyymmdd/yyyymmdd.xvw of the date you want to analyze

6.2 Analysis of Appls.dat and sqls.dat

The TOT.dat file contains the appls and sqls information executed in the pertinent day. You need to export information from this file.

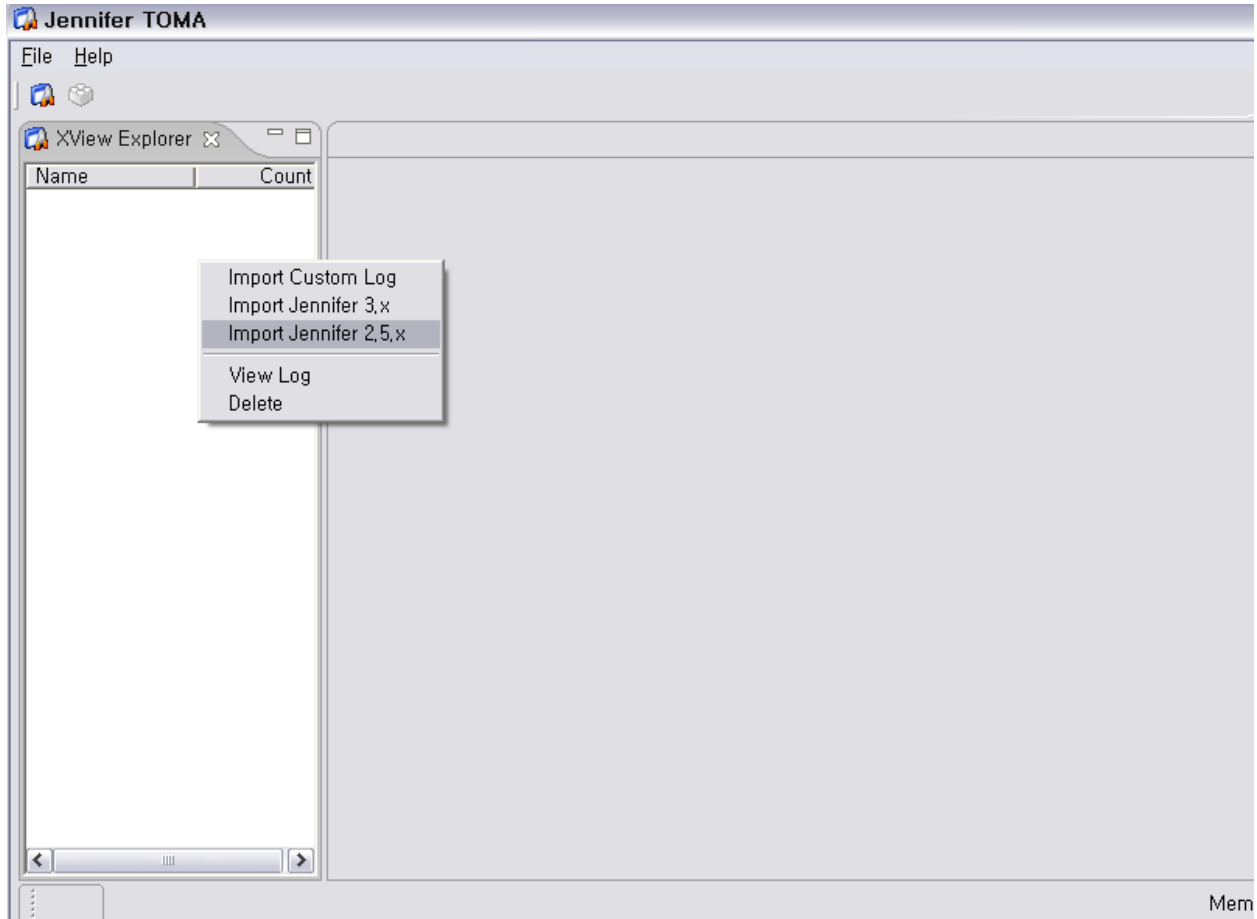
6.2.1 Execution Process

1. Copy the TOT.dat file to JenniferToolBox/addin/.
2. Execute Export25.bat [DATAFILE] [TARGET DIR]: ex) export25.bat TOT.dat.

If you follow the execution process, the appls.dat and sqls.dat files are created in the current directory.

6.3 Importing in Toma

Press the right mouse button in the XviewExplorer, and select “Import Jennifer 2.5.x.”



Import

Enter the following information in the input window:



Files

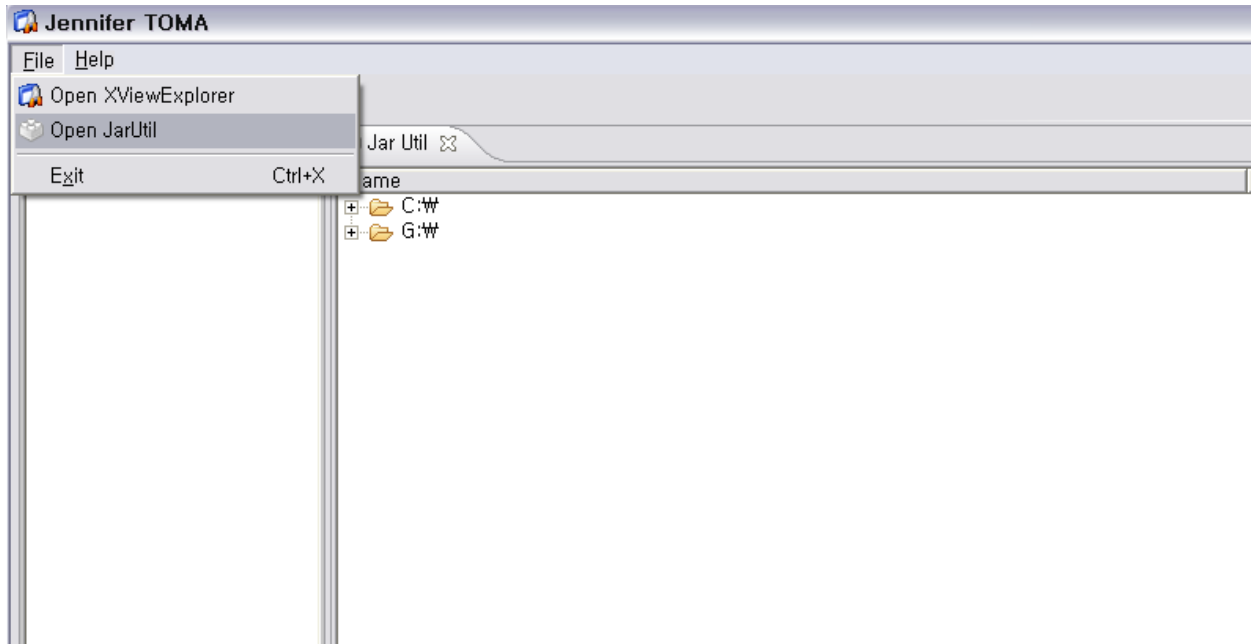


Note

You cannot see the method information in the profile contents because the server does not save the method information in Jennifer 2.5.

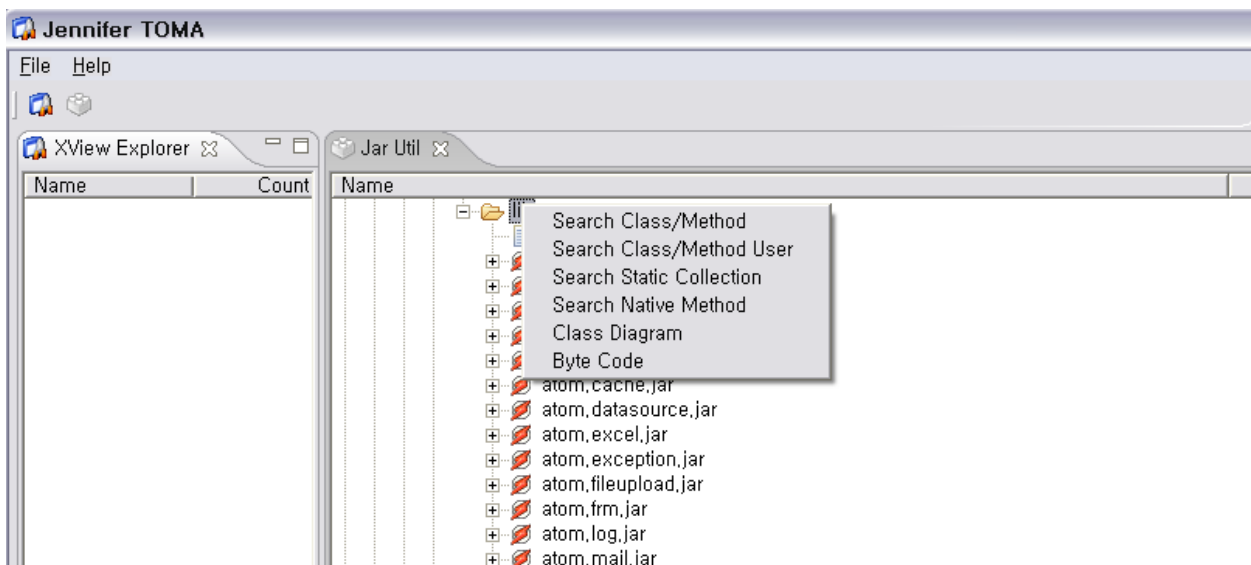
7. Using the JarUtil

The JarUtil provides a function to search information in a jar file.



Jarutil

You can use this function by selecting "Open JarUtil" from the menu.

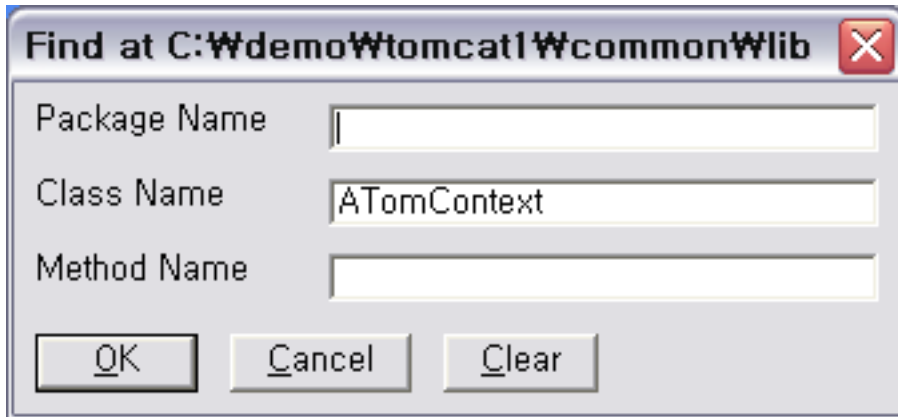


Search

If you select the directory including the jar files and click the right mouse button, you can see the various search menus starting with the title of Search.

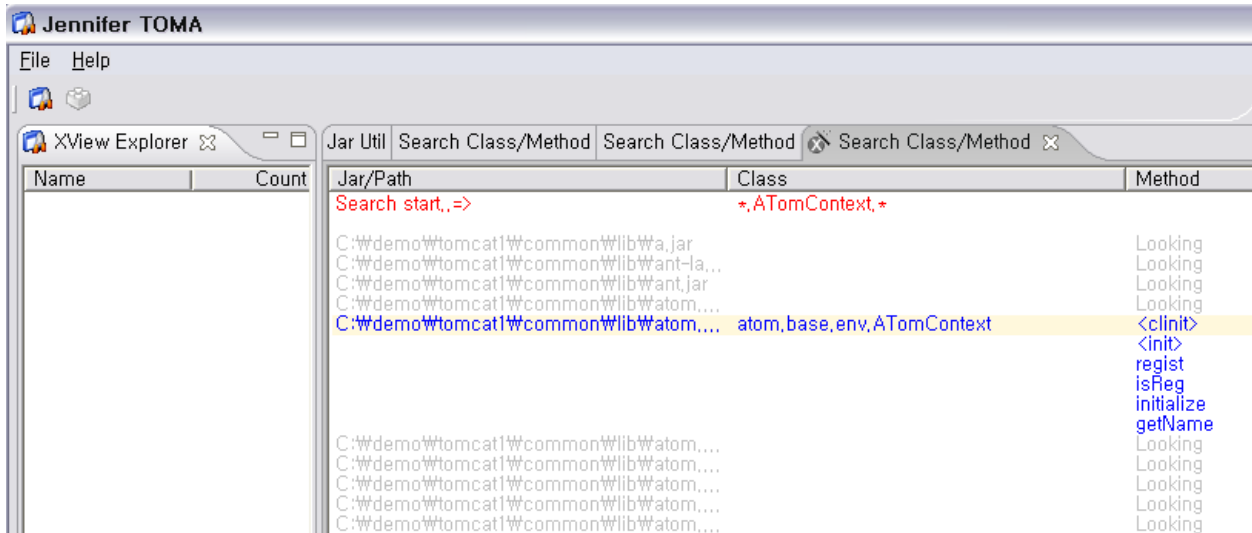
7.1 Search Class/Method

Use this function when you want to search a specific class or method in a jar file.



Search

If you click “OK,” the designated class or method is searched. In the below figure, the blue part is the search results.



Result

7.2 Search Class/Method User

Use this function when you want to search the class(es) which use(s) the designated class/method. Like the following figure, you can use this function to search the class(es) calling a specific class/method like System.gc().

Jar/Path	Class#Field	Type	Class...
Search start.			
C:\demo\tomcat1\wcommon\lib\wa...			
C:\demo\tomcat1\wcommon\lib\wa...			
C:\demo\tomcat1\wcommon\lib\wa...			
C:\demo\tomcat1\wcommon\lib\wa...	org.apache.tools.ant.DirectoryScann...	Ljava/util/Vector;	public
C:\demo\tomcat1\wcommon\lib\wa...	org.apache.tools.ant.IntrospectionHe...	Ljava/util/Hashtable;	public
C:\demo\tomcat1\wcommon\lib\wa...	org.apache.tools.ant.IntrospectionHe...	Ljava/util/Hashtable;	public
C:\demo\tomcat1\wcommon\lib\wa...	org.apache.tools.ant.loader.AntClas...	Ljava/util/Map;	public
C:\demo\tomcat1\wcommon\lib\wa...	org.apache.tools.ant.taskdefs.Execu...	Ljava/util/Vector;	public
C:\demo\tomcat1\wcommon\lib\wa...	org.apache.tools.ant.taskdefs.JDBC...	Ljava/util/Hashtable;	public
C:\demo\tomcat1\wcommon\lib\wa...	org.apache.tools.ant.taskdefs.Recor...	Ljava/util/Hashtable;	public
C:\demo\tomcat1\wcommon\lib\wa...	org.apache.tools.ant.types.Path_syst...	Ljava/util/Vector;	public
C:\demo\tomcat1\wcommon\lib\wa...	org.apache.tools.ant.util.JavaEnvUtil...	Ljava/util/Vector;	public
C:\demo\tomcat1\wcommon\lib\wa...	org.apache.tools.zip.ExtraFieldUtils.i...	Ljava/util/Hashtable;	public
C:\demo\tomcat1\wcommon\lib\wa...			
C:\demo\tomcat1\wcommon\lib\wa...	atom.base.env.XScheduler.schSet	Ljava/util/HashSet;	public
C:\demo\tomcat1\wcommon\lib\wa...	atom.base.env.ATomContext.ctx	Ljava/util/HashMap;	public
C:\demo\tomcat1\wcommon\lib\wa...	atom.util.async.TimeLimitCall.childT...	Ljava/util/HashMap;	public
C:\demo\tomcat1\wcommon\lib\wa...	atom.util.async.ThreadMgr.readyPool	Ljava/util/LinkedList;	public
C:\demo\tomcat1\wcommon\lib\wa...	atom.util.async.ThreadMgr.activePool	Ljava/util/Hashtable;	public
C:\demo\tomcat1\wcommon\lib\wa...			
C:\demo\tomcat1\wcommon\lib\wa...	atom.cache.CacheEngine.cacheHas...	Ljava/util/Hashtable;	public
C:\demo\tomcat1\wcommon\lib\wa...			
C:\demo\tomcat1\wcommon\lib\wa...	atom.datasource.DataSourcePool.si...	Ljava/util/HashMap;	public
C:\demo\tomcat1\wcommon\lib\wa...	atom.datasource.DataSourcePool.si...	Ljava/util/HashMap;	public
C:\demo\tomcat1\wcommon\lib\wa...			
C:\demo\tomcat1\wcommon\lib\wa...			
C:\demo\tomcat1\wcommon\lib\wa...			
C:\demo\tomcat1\wcommon\lib\wa...			
C:\demo\tomcat1\wcommon\lib\wa...	atom.frm.ejb.EJBHomeFactory.aFact...	Ljava/util/Map;	public
C:\demo\tomcat1\wcommon\lib\wa...			

Result

7.4 Search Native Method

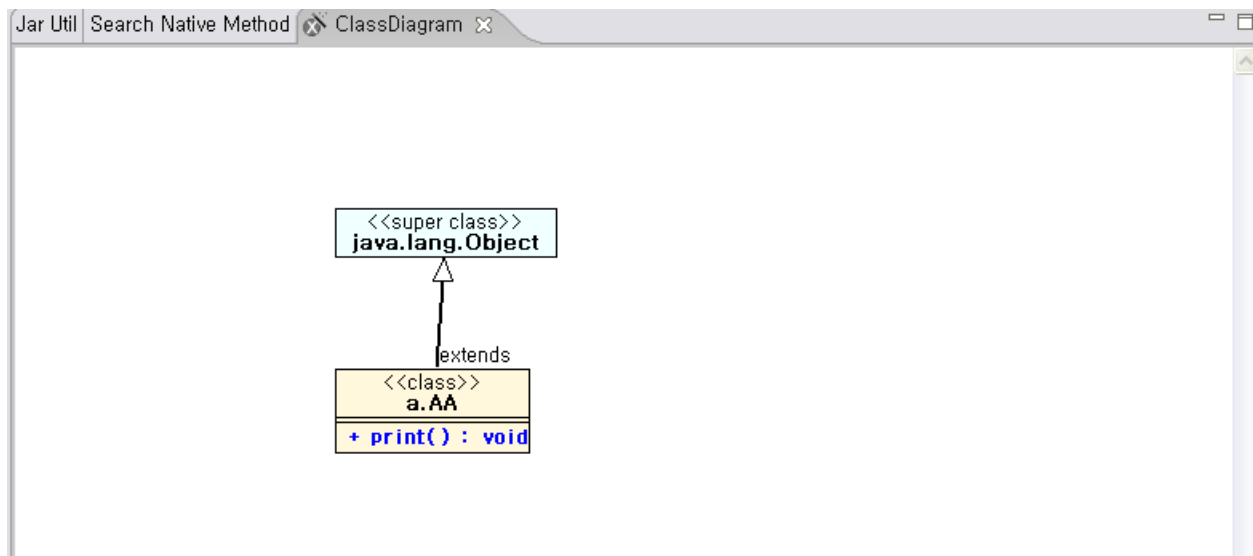
This function is used to search the native methods. Especially, it is used effectively to identify the suspicious module(s) when the total Java process memory usage increases.

7.5 Class Diagram

You can use this function when you draw the class diagram. If you select a class in the jar file and choose “Class Diagram” from the menu that appears when you click the right mouse button, the class diagram image is shown as follows:

Name	Size	Time
common		
classes		
endorsed		
lib		
SetCharacterEncodingFilter.java	8,853	2004-07-20 11:03:
a.jar	3,473	2007-03-09 16:44:
a/AA class	416	2007-03-09 16:44:
a/BI	1,168	2007-03-09 16:44:
a/CI	1,067	2007-03-09 16:44:
a/S	780	2007-03-09 16:44:
a/Te	1,004	2007-03-09 16:44:
a_servic	97,690	2004-06-18 10:00:
ant-laur	8,412	2004-06-18 10:00:
ant.jar	958,858	2004-06-18 10:00:

Menu

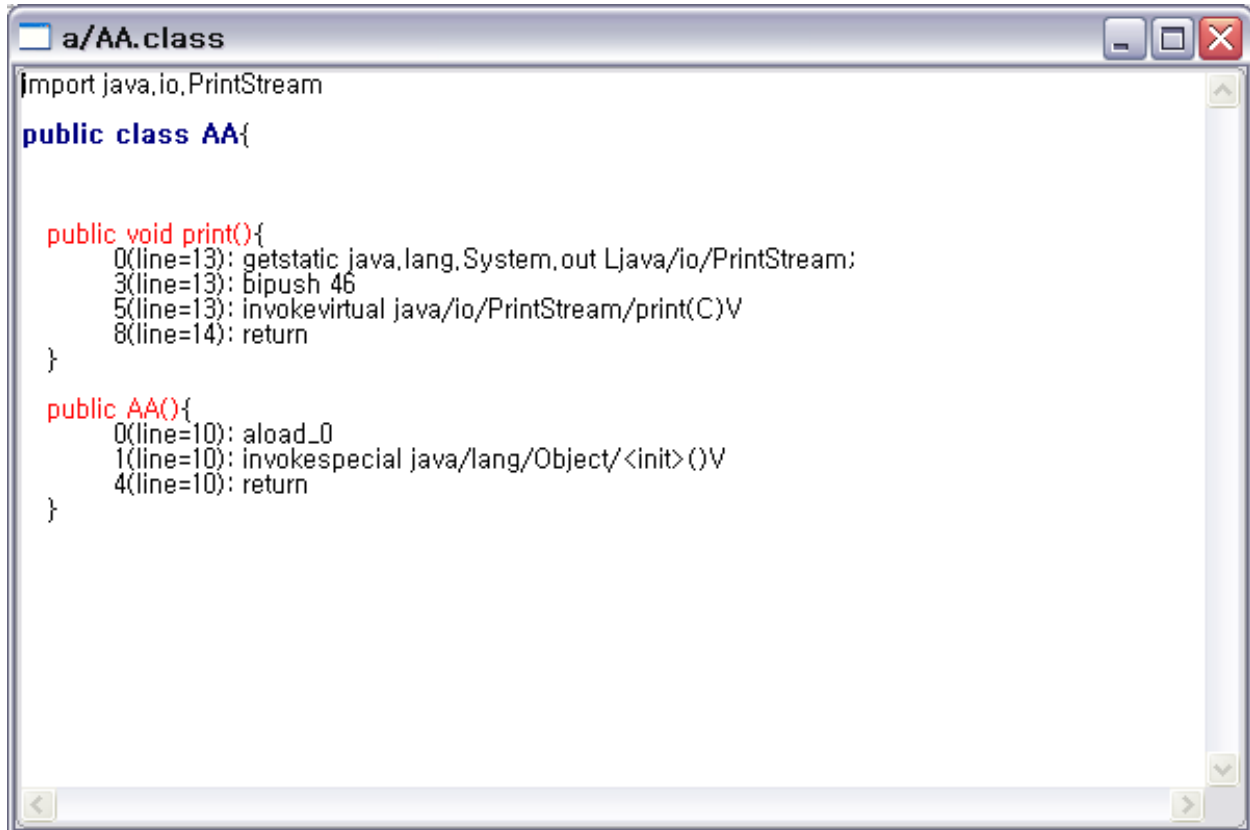


Class Diagram

7.6 Byte Code

You can check the contents of a Java class' byte code almost at the level of the decompiled source. The "Byte Code" menu is also used when you check the signatures such as method and member variables.

Select a class and click the right mouse button. From the menus you see, select "Byte Code" and you will see the information as shown in the following figure:



```
import java.io.PrintStream

public class AA{

    public void print(){
        0(line=13): getstatic java.lang.System.out Ljava/io/PrintStream;
        3(line=13): bipush 46
        5(line=13): invokevirtual java/io/PrintStream/print(C)V
        8(line=14): return
    }

    public AA(){
        0(line=10): aload_0
        1(line=10): invokespecial java/lang/Object/<init>()V
        4(line=10): return
    }
}
```

Byte Code