Atom Developer's Guide

Whilst all reasonable care has been taken to ensure that the details are true and not misleading at the time of publication, no liability whatsoever is assumed by Automature LLC, or any supplier of Automature LLC, with respect to the accuracy or any use of the information provided herein.

Any license, delivery and support of software require entering into separate agreements with Automature LLC.

This document may contain confidential information and may not be modified or reproduced, in whole or in part, or transmitted in any form to any third party, without the written approval from Automature LLC.

Revision History

Ver.	Date	Name	Description of Changes
1.0	01/06/12	Dipayan Sengupta	First Edition
1.1	15/06/12	Dipayan Sengupta	Second Edition
1.2	21/06/12	Dipayan Sengupta	Revised Edition

Contents

1 Introduction...4

1.1 Document Purpose..4

1.2 Intended Audience..4

1.3 References and Other Related Documents..............4

2 Dependencies..5

2.1 Zug SDK distribution:.......................................5

3 Writing C# in-process Atoms..6

3.1 Creating a new project.......................................6

3.2 Accessing ZugLib methods..................................7

3.3 Configuring the Assembly Information....................8

3.4 Writing a Class File 9

3.5 Architectural Mechanism for calling C# dll atoms....10

3.5 Accessing ZugLib methods.................................10

3.5.1 Accessing Context Variable 10

3.5.2 Writing Debug Messages...................................10

3.5.3 Exception Handling..10

3.6 Writing a simple atom.......................................11

3.7 Build the project..11

3.8 Register the dll..11

3.9 Configuring the ZugINI.xml file...........................12

3.10 Writing a simple test case................................13

4 Writing Java in-process atoms...14

4.1 Create a project...14

4.2 Accessing the ZugLib methods............................15

4.3 Writing a Class File..16

4.4 Architectural Mechanism for calling Java atoms......17

4.5 Accessing ZugLib methods.................................17

4.5.1 Accessing Context Variable 17

4.5.2 Writing Debug Messages...................................17

4.5.3 Exception Handling..17

4.6 Writing a simple atom.......................................18

4.7 Build the project..18

4.8 Configuring the ZugINI.xml file...........................19

4.9 Writing a simple test case..................................19

1 Introduction

- Atoms in Zug, are the basic reusable units of work. They are entities, such as functions, procedures, programs, or scripts that can be executed either within the context of a main program, or as a standalone program at the command line level in a shell (e.g. the Command Prompt in Windows).

- Atoms are intended to be generic enough, so that they can be applied in test case design in a range of situations. When executing as a standalone program, they are called *out-of-process* atoms. When implemented as a function or method, they are referred to as *in-process* atoms, since they can be invoked within the process context of the execution engine.

- Atoms allow users of Zug to extend the functionality of the engine, to apply it to testing different kinds of applications in new domains.

1.1 Document Purpose

This document provides detailed information about developing in-process atoms in various languages e.g. Java, C, C++, C#.

1.2 Intended Audience

This Reference Manual is intended for test case designers and programmers, and assumes prior knowledge of a high level language It also assumes a good understanding of the development and target platform environments.

1.3 References and Other Related Documents

The following documents provide additional useful information about Automature's other products, and how they relate to Zug.

1. Zuoz Reference Guide
2. Chur Programmer's Guide

2 Dependencies

Before running ZUG on a computer, it has to meet some basic requirements:

Operating System – Microsoft Windows XP or higher

Software – **Java(TM) SE Runtime Environment 1.6**

NetBeans IDE 7.0

Microsoft Visual Studio 2008 or higher.

2.1 Zug SDK distribution:

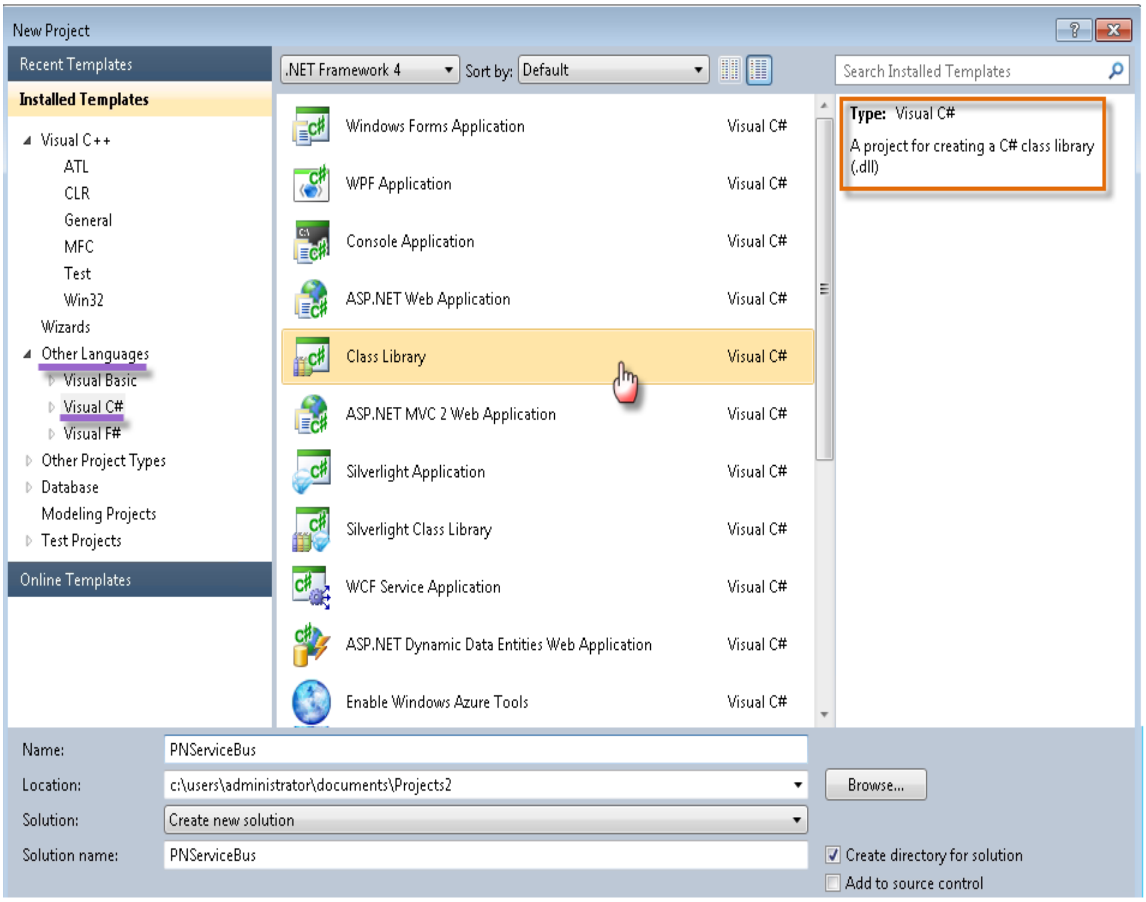
Zug SDK is distributed with the standard Zug kit. However, one does not require a run-time license to use it as an SDK. For testing purposes however, it is recommended to ask Automature for a “free” developer license.

3 Writing C# in-process Atoms

Kindly follow the following steps :

3.1 Creating a new project

Create a new project in Visual Studio. (Ex.- DemoProject)


New Project
Recent Templates
Installed Templates
Visual C++
ATL
CLR
General
MFC
Test
Win32
Wizards
Other Languages
Visual Basic
Visual C#
Visual F#
Other Project Types
Database
Modeling Projects
Test Projects
Online Templates
Search Installed Templates
Windows Forms Application Visual C#
WPF Application Visual C#
Console Application Visual C#
ASP.NET Web Application Visual C#
Class Library Visual C#
ASP.NET MVC 2 Web Application Visual C#
Silverlight Application Visual C#
Silverlight Class Library Visual C#
WCF Service Application Visual C#
ASP.NET Dynamic Data Entities Web Application Visual C#
Enable Windows Azure Tools Visual C#
Browse...

.NET Framework 4 Sort by: Default

Type: Visual C#

A project for creating a C# class library (.dll)

Name: PNServiceBus

Location: c:\users\administrator\documents\Projects2

Solution: Create new solution

Solution name: PNServiceBus

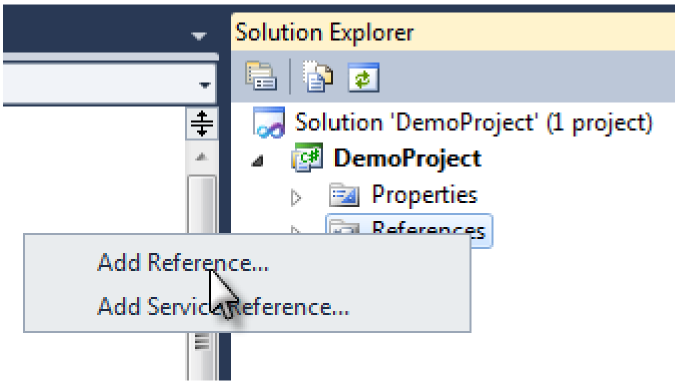
☑ Create directory for solution

□ Add to source control

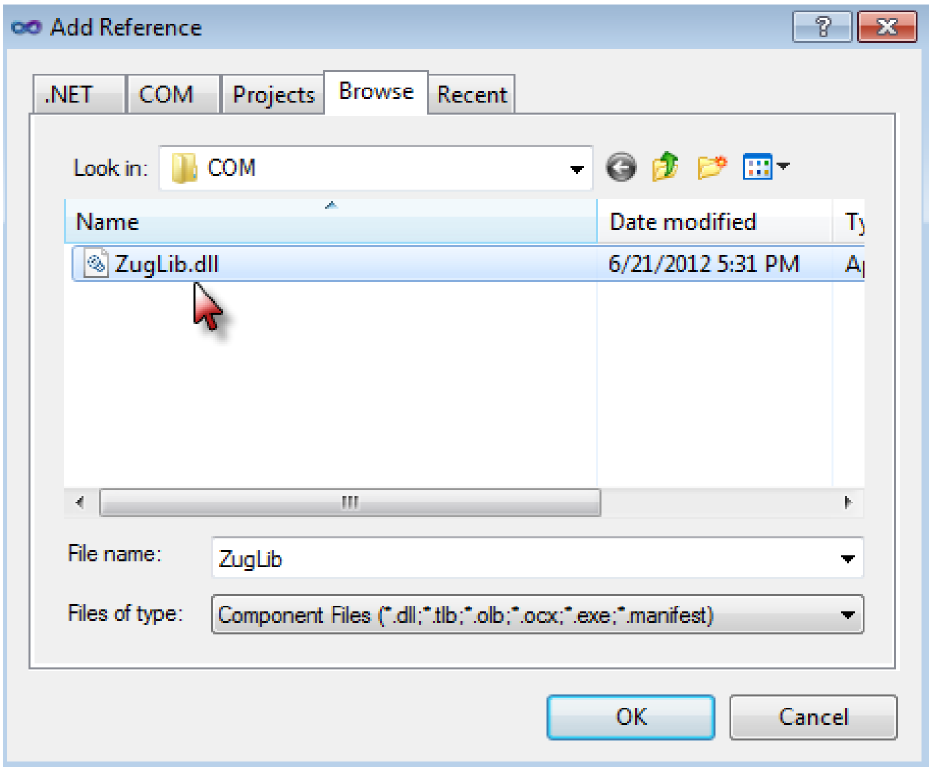
3.2 Accessing ZugLib methods

You have to add a reference to the ZugLib.dll (located in the SDK folder in Zug installation directory) to your custom C# project.
This will expose the ZugLib methods in your project.

Right Click on the References option in the Solution Explorer.



Browse to the Zug installation directory, Navigate to SDK/COM and you will find ZugLib.dll.



Look in: COM

Name	Date modified	Ty
ZugLib.dll	6/21/2012 5:31 PM	A

File name: ZugLib

Files of type: Component Files (*.dll;*.tlb;*.olb;*.ocx;*.exe;*.manifest)

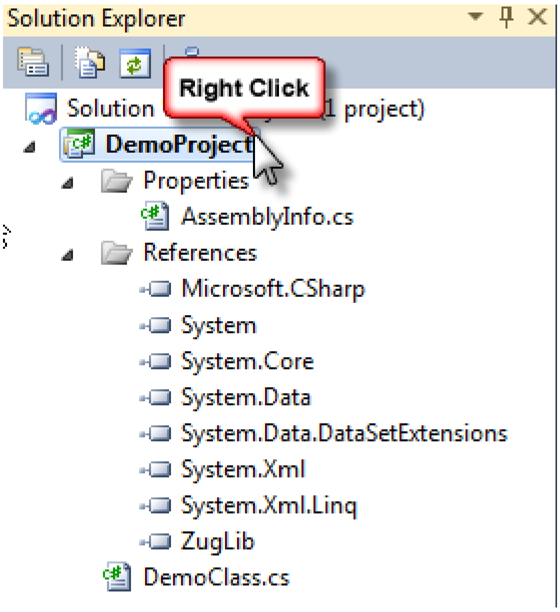
3.3 Configuring the Assembly Information

You need to give the assembly information before writing the class file.

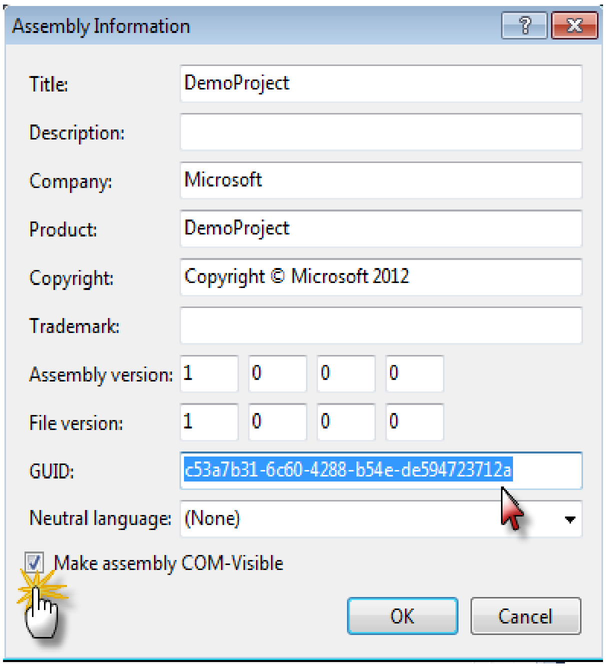
```
[ComVisible(true), Guid("b33b95fb-b977-4e2c-836a-64d33da85382"),
ClassInterface(ClassInterfaceType.None),
ComSourceInterfaces(typeof(IZugAtomExecutor))]
ProgId("Automature.DemoProject")]
```

You need to configure the GUID and also set the ProgID before you write the class file.

Right Click on the Project Name and Select Properties.



Click on Assembly Language. Select the check box stating Make assembly COM-visible and copy the GUID and replace it in the code stated above.


Assembly Information

Title:	DemoProject
Description:	
Company:	Microsoft
Product:	DemoProject
Copyright:	Copyright © Microsoft 2012
Trademark:	
Assembly version:	1 0 0 0
File version:	1 0 0 0
GUID:	c53a7b31-6c60-4288-b54e-de594723712a
Neutral language:	(None)

☑ Make assembly COM-Visible

3.4 Writing a Class File

You will need to import the following libraries additionally as mentioned below:

```
using System.Runtime.InteropServices;
using System.Reflection;
using Automature.Zug;
```

You will have to implement the dispatch method inside your custom class. The signature of the Dispatch method is as follows:

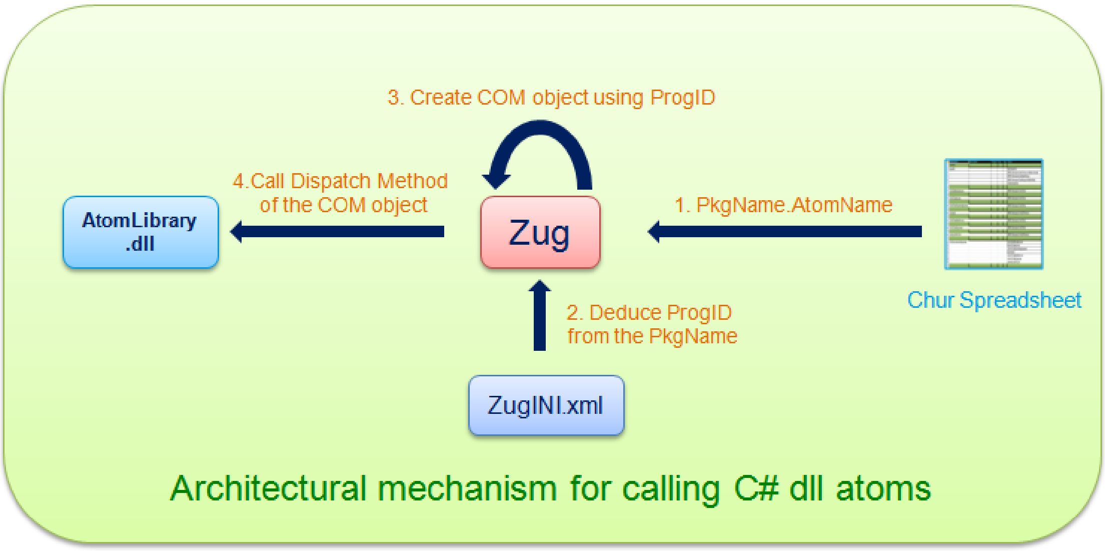
```
 int dispatch(string method_name, String inputs);
```

Following is the dispatch method:

```
public class DemoClass : IZugAtomExecutor
    {
      public void dispatch(string method_name, string inputs)
       ParameterInfo[] args;
       bool method_found_flag = false;
       bool arg_match_flag=false;
       try
          {
            PNServiceBus iClass = new PNServiceBus();
            Type PNSBObj = iClass.GetType();
            MethodInfo[] iMethods = PNSBObj.GetMethods();
            Object[] iParams = ZugLib.DelimitArg(inputs);
            foreach (MethodInfo lMethod in iMethods)
               {
              if (lMethod.Name.ToUpper().Equals(method_name.ToUpper()) )
                  {
                     method_found_flag = true;
                     args = lMethod.GetParameters();
                     if (iParams.Length == args.Length)
                         {
                            arg_match_flag = true;
                            lMethod.Invoke(iClass, iParams);
                            break;
                         }
                   }
                }

              if (!method_found_flag)
              {
               throw new Exception("Method not found : " + method_name);
              }
             if (!arg_match_flag)
              {
              throw new Exception("Argument mismatch : " + method_name);
              }
           }
```

```
         catch(Exception e)
          {
           Console.WriteLine("Message from dispatch ::" +
e.InnerException);  prints the root cause just as in getCause in Java
          ZugLib.Log("error", "error occured while invoking dispatch");
          throw e;
          }
    }
```

3.5 Architectural Mechanism for calling C# dll atoms


3. Create COM object using ProgID
4.Call Dispatch Method of the COM object
AtomLibrary .dll
Zug
1. PkgName.AtomName
Chur Spreadsheet
2. Deduce ProgID from the PkgName
ZugINI.xml
Architectural mechanism for calling C# dll atoms

3.5 Accessing ZugLib methods

3.5.1 Accessing Context Variable

In order to access context variable methods from atoms you need to write

ZugLib.SetContextVar(contextvar name, contextvar value)

3.5.2 Writing Debug Messages

In order to write debug messages from atoms you need to write
ZugLib.("loglevel".message)

This "loglevel" can be "error" or "debug" or "info".

3.5.3 Exception Handling

When an exception occurs in a C# atom it can be handled using the Try and Catch block. The exception should be thrown back.
Please refer to the Section 3.6 which illustrates an example of how to handle Exceptions.

3.6 Writing a simple atom

The following is an example of a simple atom which takes two inputs as text and compares them. It also shows how to handle Exceptions by using the Try and Catch Block.

```
    public void Compare(string First_input, string Second_input)
        {
          try
            {

           if (string.Equals(First_input, Second_input,
StringComparison.OrdinalIgnoreCase))
                {
                    Console.WriteLine("Strings matched successfully");
                }
                else
                {
                    ZugLib.Log("error", "Strings do not match");
                    throw new Exception("Strings do not match");
                }
            }

            catch (Exception e)
            {
                ZugLib.Log("error", "error occurred while invoking
Compare");
                throw e;
            }

        }
```

3.7 Build the project

Build the project and DemoProject.dll file is created in the project output directory
e.g. -
C:\Users\Administrator\Documents\Projects\DemoProject\DemoProject\bin\Debug\DemoProject.dll

3.8 Register the dll

Open the Command Prompt as Administrator and run the following command:

```
%windir%\Microsoft.NET\Framework\v<framework_version>2.0.50727\RegAsm.exe
/verbose /nologo /codebase "<path of the dll file>"
```

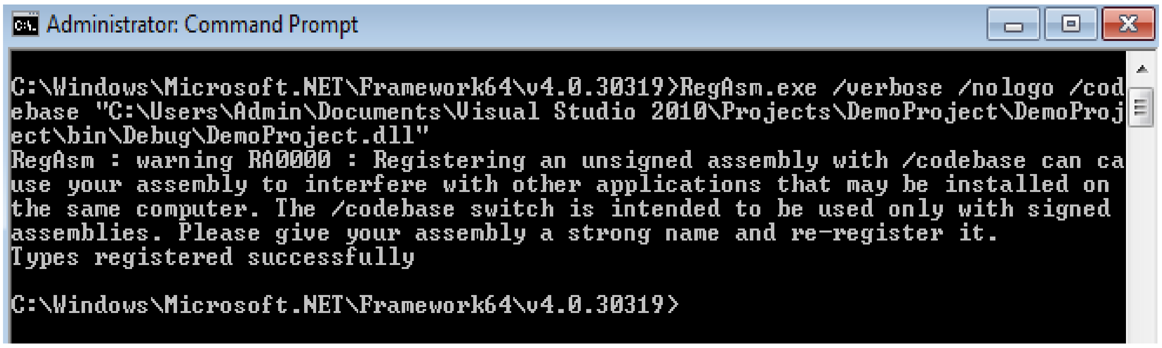
For 64-bit Operating System, rewrite the same as

```
%windir%\Microsoft.NET\Framework64\v<framework_version>2.0.50727\RegAsm.exe
/verbose /nologo /codebase "<path of the dll file>"
```

Example:

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319>RegAsm.exe /verbose
/nologo /codebase "C:\Users\Admin\Documents\Visual Studio
2010\Projects\DemoProject\DemoProject\bin\Debug\DemoProject.dll"
```

The output will be as follows:



```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319>RegAsm.exe /verbose /nologo /cod
ebase "C:\Users\Admin\Documents\Visual Studio 2010\Projects\DemoProject\DemoProj
ect\bin\Debug\DemoProject.dll"
RegAsm : warning RA0000 : Registering an unsigned assembly with /codebase can ca
use your assembly to interfere with other applications that may be installed on
the same computer. The /codebase switch is intended to be used only with signed
assemblies. Please give your assembly a strong name and re-register it.
Types registered successfully

C:\Windows\Microsoft.NET\Framework64\v4.0.30319>
```

3.9 Configuring the ZugINI.xml file

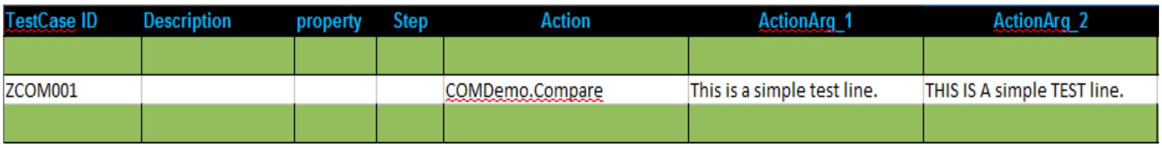
Inside the Zug Installation directory you will find the ZugINI.xml file.

Please add the following inside <inprocesspackages> tag :

```
<inprocesspackage name="COMDemo" language="COM">

        <prog-id>Automature.DemoProject</prog-id>

</inprocesspackage>
```

3.10 Writing a simple test case

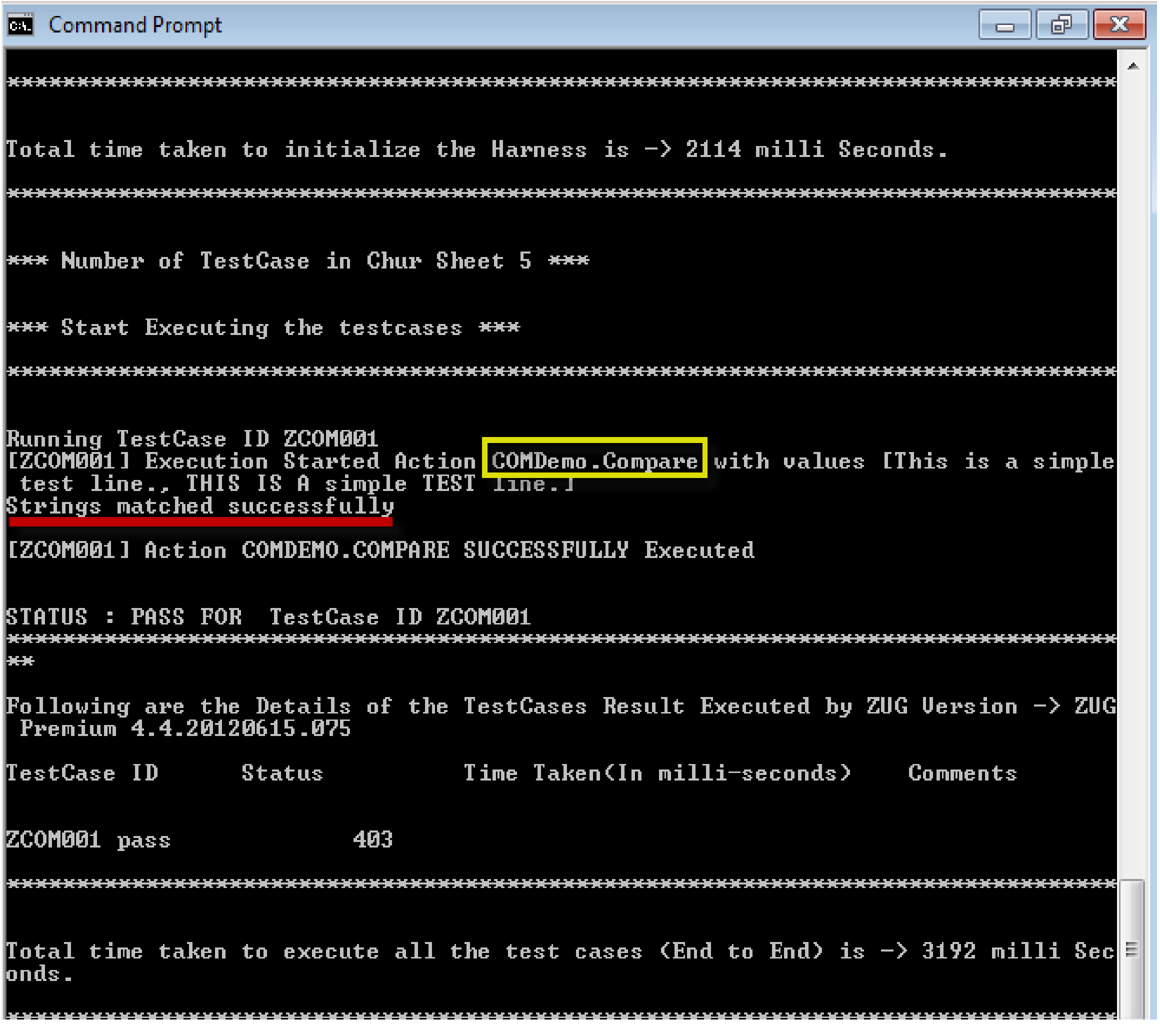
The following is a simple test case written in Chur. The atom simply compares two strings.



TestCase ID	Description	property	Step	Action	ActionArg_1	ActionArg_2
ZCOM001				COMDemo.Compare	This is a simple test line.	THIS IS A simple TEST line.

For more information on writing test cases in Chur, please refer to **Chur Programmer's Guide.**

The output of the test case is as follows:

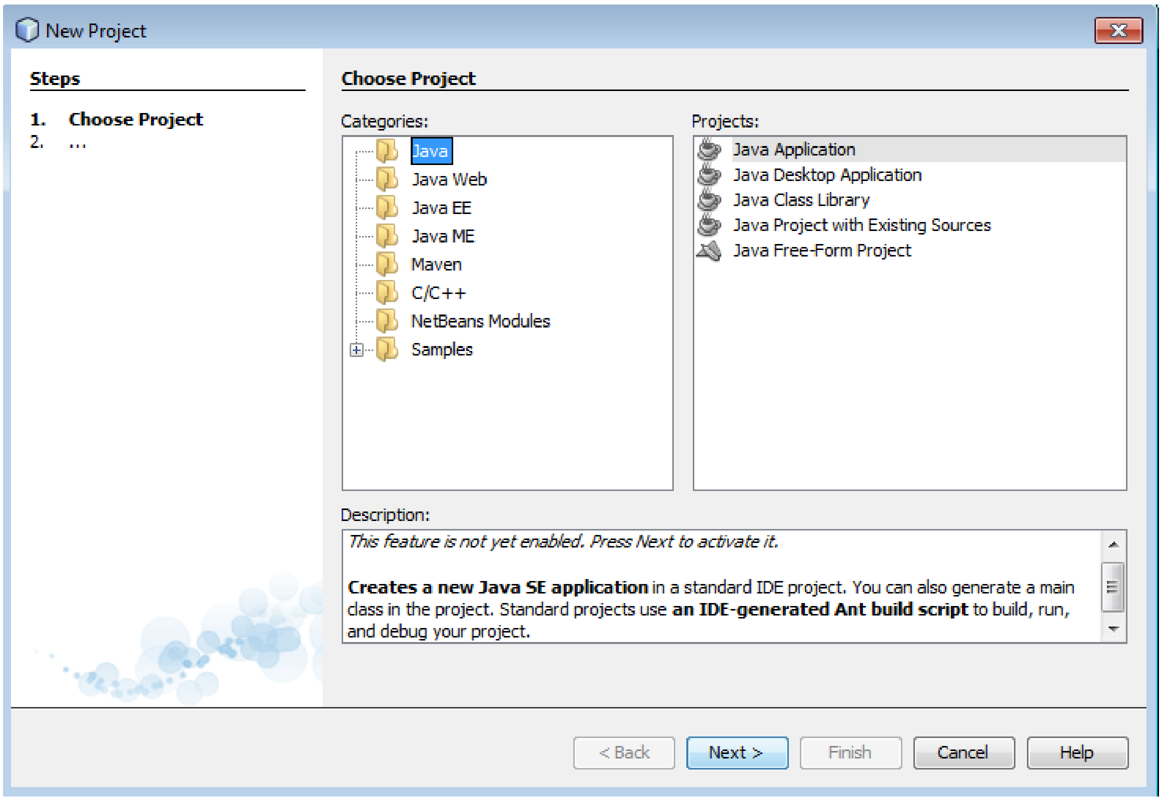


```
********************************************************************************

Total time taken to initialize the Harness is -> 2114 milli Seconds.

********************************************************************************

*** Number of TestCase in Chur Sheet 5 ***

*** Start Executing the testcases ***

********************************************************************************

Running TestCase ID ZCOM001
[ZCOM001] Execution Started Action COMDemo.Compare with values [This is a simple
 test line., THIS IS A simple TEST line.]
Strings matched successfully

[ZCOM001] Action COMDEMO.COMPARE SUCCESSFULLY Executed

STATUS : PASS FOR  TestCase ID ZCOM001
********************************************************************************
**

Following are the Details of the TestCases Result Executed by ZUG Version -> ZUG
 Premium 4.4.20120615.075

TestCase ID      Status          Time Taken(In milli-seconds)    Comments

ZCOM001 pass              403

********************************************************************************

Total time taken to execute all the test cases (End to End) is -> 3192 milli Sec
onds.

********************************************************************************
```

4 Writing Java in-process atoms

4.1 Create a project

Create a project in NetBeans. (Example – JavaDemo)



New Project

Steps

1. **Choose Project**
2. ...

Choose Project

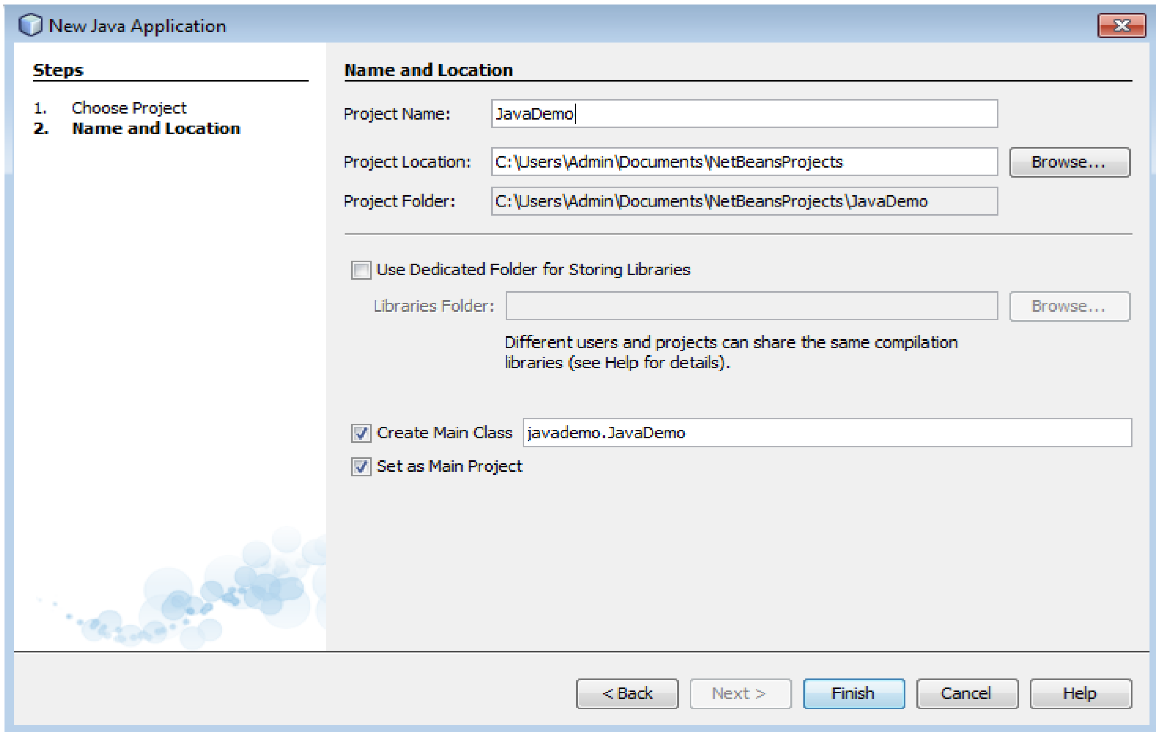
Categories: Java, Java Web, Java EE, Java ME, Maven, C/C++, NetBeans Modules, Samples

Projects: Java Application, Java Desktop Application, Java Class Library, Java Project with Existing Sources, Java Free-Form Project

Description:

This feature is not yet enabled. Press Next to activate it.

Creates a new Java SE application in a standard IDE project. You can also generate a main class in the project. Standard projects use **an IDE-generated Ant build script** to build, run, and debug your project.



New Java Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name: JavaDemo

Project Location: C:\Users\Admin\Documents\NetBeansProjects

Project Folder: C:\Users\Admin\Documents\NetBeansProjects\JavaDemo

□ Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

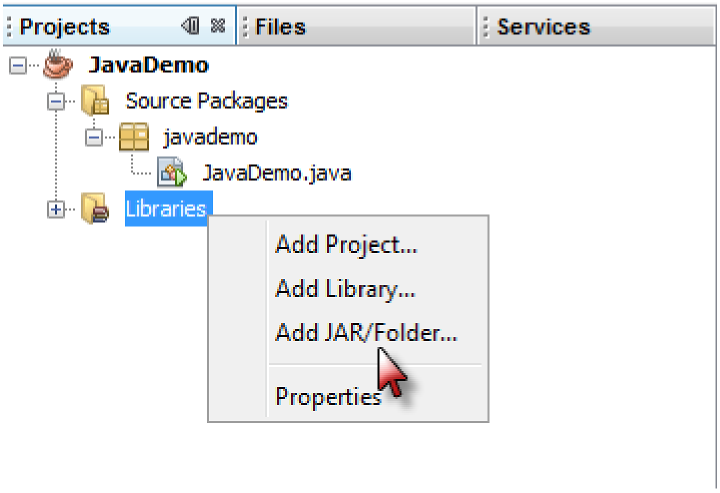
☑ Create Main Class javademo.JavaDemo

☑ Set as Main Project

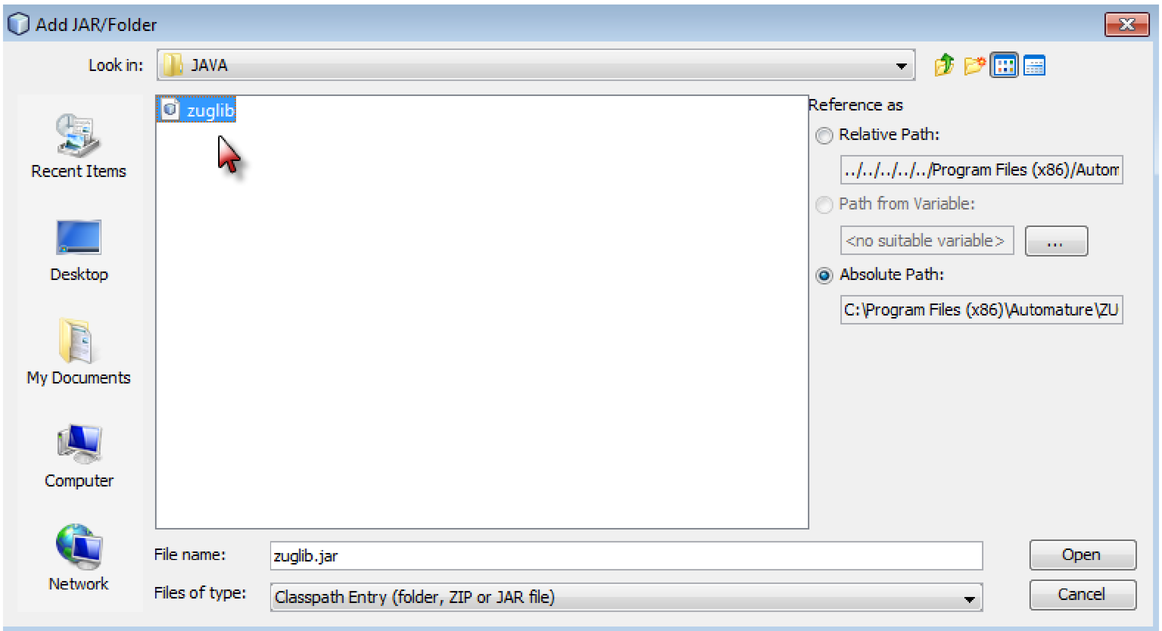
4.2 Accessing the ZugLib methods

You need to add **zuglib.jar** file to your custom Java project.

Right Click on Libraries and add Jar Folder.



Navigate to the Zug installation Directory where you will find SDK folder. Within the SDK folder, you will find JAVA folder which contains zuglib.jar.



Look in: JAVA

Reference as

○ Relative Path:

../../../../../Program Files (x86)/Autom

○ Path from Variable:

<no suitable variable>

● Absolute Path:

C:\Program Files (x86)\Automature\ZU

File name: zuglib.jar

Files of type: Classpath Entry (folder, ZIP or JAR file)

4.3 Writing a Class File

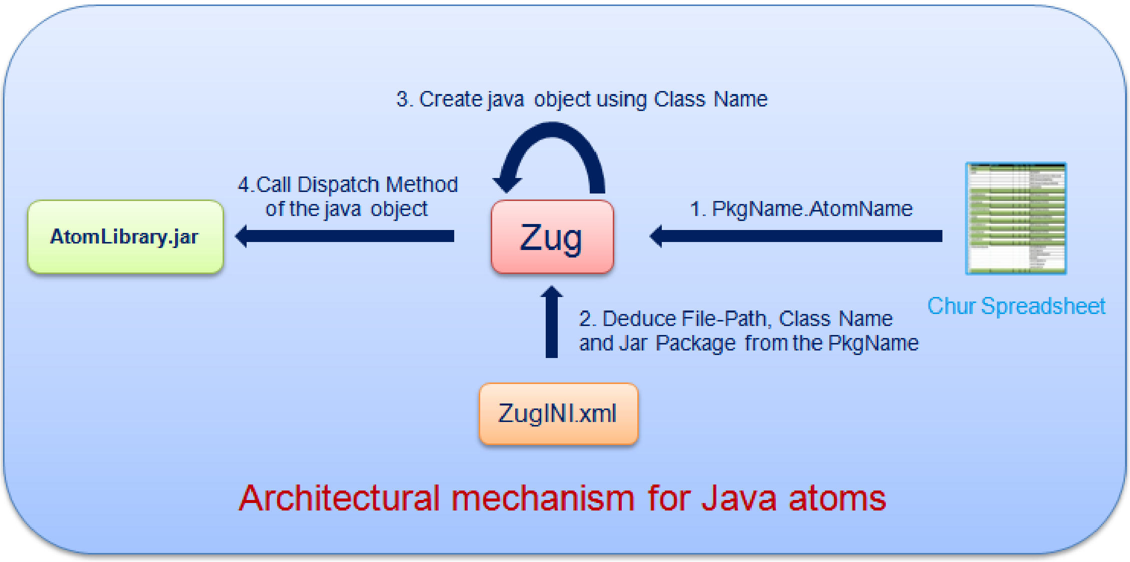
You need to import the following packages:

```
import com.automature.zug.exceptions.AtomExecutionException;
import com.automature.zug.exceptions.AtomNotFoundException;
import java.lang.reflect.Method;
import com.automature.zug.lib.AtomExecutor;
import com.automature.zug.lib.ZugLib;
import java.lang.reflect.InvocationTargetException;
import java.util.ArrayList;
```

You will have to implement the dispatch method inside your custom class. Following is the dispatch method:

```
public class JavaDemo implements AtomExecutor{

public void dispatch(String method_name, ArrayList<String> inputs)
throws AtomExecutionException, AtomNotFoundException {
        boolean method_found_flag = false, arg_match_flag=false;
        try {
            Class<?> iClass = Class.forName(this.getClass().getName());
            Method[] iMethods = iClass.getMethods();
            Object iObject = iClass.newInstance();
            Object iParams[] = inputs.toArray();
            method_found_flag = false;
            for (Method lMethod : iMethods) {
              if (lMethod.getName().equalsIgnoreCase(method_name)) {
                        method_found_flag = true;
 if((lMethod.getParameterTypes().length == inputs.toArray().length) ||
(inputs.isEmpty() && lMethod.getParameterTypes().length==0)){
                                    arg_match_flag = true;
                                    lMethod.invoke(iObject,iParams);
                                    break;
                              }
                        }
                  }
                  if (method_found_flag && !arg_match_flag) {
                  throw new AtomNotFoundException("Argument mismatch");
                  }
            if (method_found_flag == false) {
      throw new AtomNotFoundException(method_name);
                           }
                  }
            catch (InvocationTargetException It) {
                  throw new
AtomExecutionException("StringOperation.dispatch\n", It.getCause());
            }
            catch (Exception e) {
                  throw new
AtomExecutionException("StringOperation.dispatch\n"+e.getClass(), e);
                        }
     }
   }
```

4.4 Architectural Mechanism for calling Java atoms


3. Create java object using Class Name
4.Call Dispatch Method of the java object
AtomLibrary.jar
Zug
1. PkgName.AtomName
Chur Spreadsheet
2. Deduce File-Path, Class Name and Jar Package from the PkgName
ZugINI.xml

Architectural mechanism for Java atoms

4.5 Accessing ZugLib methods

4.5.1 Accessing Context Variable

In order to access context variable methods from atoms you need to write
ZugLib.SetContextVar(contextvar name, contextvar value);

4.5.2 Writing Debug Messages

In order to write debug messages from atoms you need to write

ZugLib.log(log_level, Message);

This “loglevel” can be “error” or “debug” or “info”.

4.5.3 Exception Handling

When an exception occurs in a Java atom it can be handled using the Try and Catch block. The exception should be thrown back.
Please refer to the Section 4.6 which illustrates an example of how to handle Exceptions.

4.6 Writing a simple atom

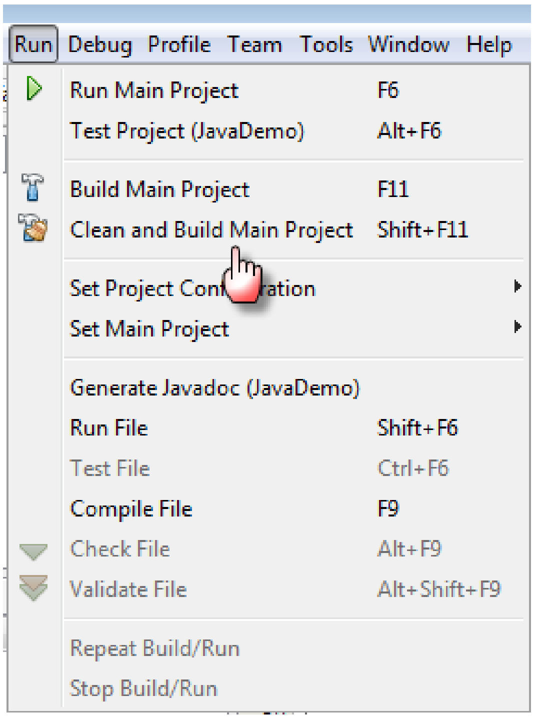
The following is an example of a simple atom which takes two input strings and compares them. It also shows how to handle Exceptions by using the Try and Catch Block.

```
 public void compare(String str1, String str2) throws
AtomExecutionException
    {
            String msg = "";
                String msg2="";
            try {
                  msg="Error while comparing two strings.";
                  if (!str1.equalsIgnoreCase(str2)) {
                        msg="Strings do not match";
                        throw new Exception("String do not match");
                  }
                 else
                 System.out.println("Strings matched Successfully");

            } catch (Exception e) {
                  ZugLib.log("error",e.getMessage());
            }
      }
```

4.7 Build the project

Build the project and JavaDemo.jar file is created in the project output directory

e.g. -`C:\Users\Admin\Documents\NetBeansProjects\JavaDemo\dist\JavaDemo.jar`


Run Debug Profile Team Tools Window Help
Run Main Project F6
Test Project (JavaDemo) Alt+F6
Build Main Project F11
Clean and Build Main Project Shift+F11
Set Project Configuration
Set Main Project
Generate Javadoc (JavaDemo)
Run File Shift+F6
Test File Ctrl+F6
Compile File F9
Check File Alt+F9
Validate File Alt+Shift+F9
Repeat Build/Run
Stop Build/Run

4.8 Configuring the ZugINI.xml file

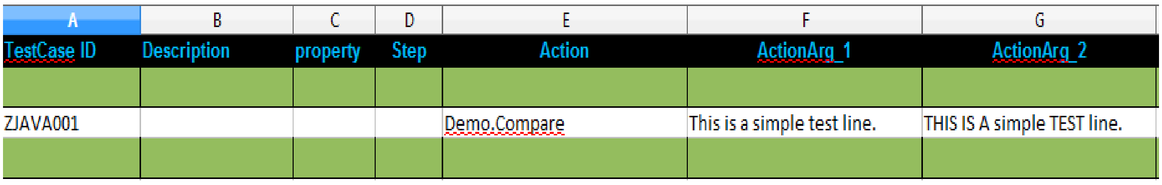
Inside the Zug Installation directory you will find the ZugINI.xml file.

Please add the following inside <inprocesspackages> tag :

```
<inprocesspackage language="java" name="Demo">
<file-path>
C:\Users\Admin\Documents\NetBeansProjects\JavaDemo\dist
</file-path>
      <jar-package>javademo</jar-package>
      <class-name>JavaDemo</class-name>
</inprocesspackage>
```

4.9 Writing a simple test case

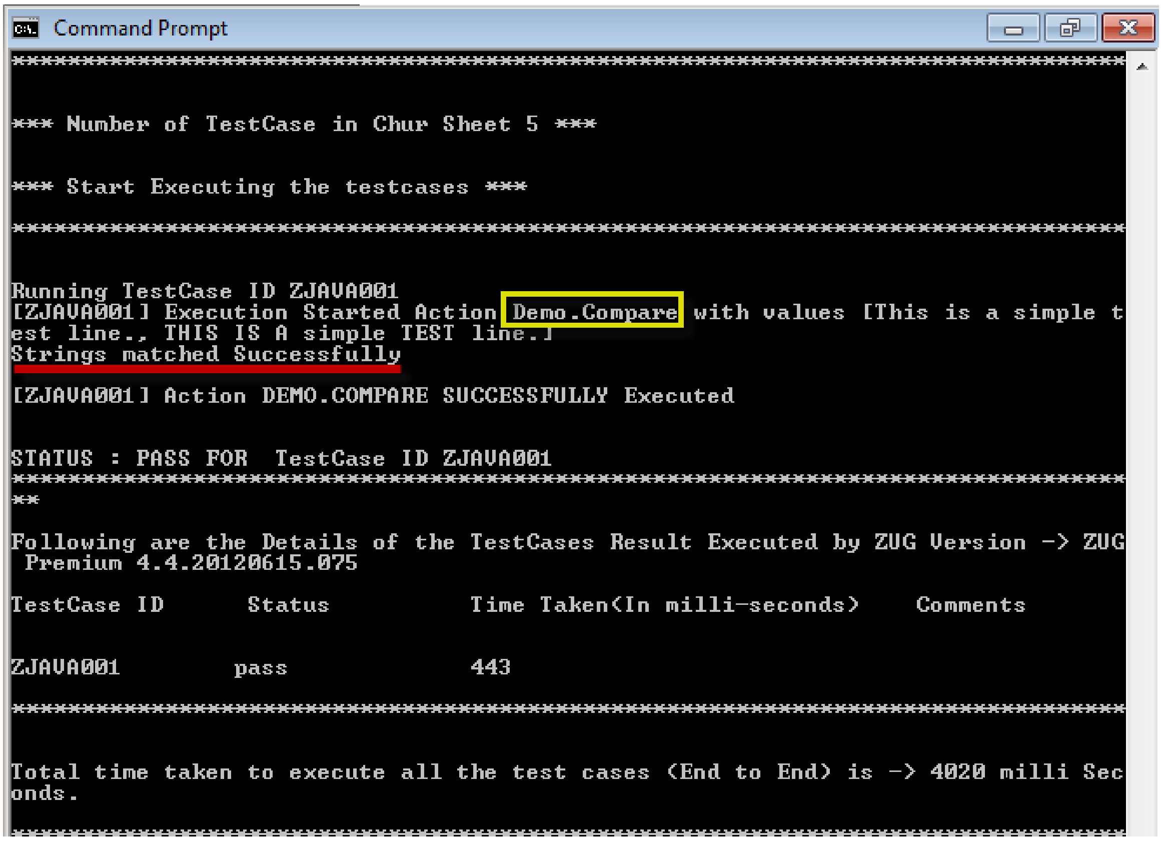
The following is a simple test case written in Chur. The atom simply compares two strings.



A	B	C	D	E	F	G
TestCase ID	Description	property	Step	Action	ActionArg_1	ActionArg_2
ZJAVA001				Demo.Compare	This is a simple test line.	THIS IS A simple TEST line.

For more information on writing test cases in Chur, please refer to **Chur Programmer's Guide.**

The output of the test case is as follows:



```
******************************************************************************

*** Number of TestCase in Chur Sheet 5 ***

*** Start Executing the testcases ***

******************************************************************************

Running TestCase ID ZJAVA001
[ZJAVA001] Execution Started Action Demo.Compare with values [This is a simple t
est line., THIS IS A simple TEST line.]
Strings matched Successfully

[ZJAVA001] Action DEMO.COMPARE SUCCESSFULLY Executed

STATUS : PASS FOR  TestCase ID ZJAVA001
******************************************************************************
**

Following are the Details of the TestCases Result Executed by ZUG Version -> ZUG
 Premium 4.4.20120615.075

TestCase ID       Status           Time Taken(In milli-seconds)    Comments

ZJAVA001         pass             443

******************************************************************************

Total time taken to execute all the test cases (End to End) is -> 4020 milli Sec
onds.

******************************************************************************
```