

Start OpenOffice automation with Eclipse and Java

Author: Lars Behrmann

Email: lb@OpenDocument4all.com

WWW: <http://www.OpenDocument4all.com>

Table of Contents

1. About.....	2
2. Setup your operating system.....	2
3. Our first OpenOffice Java application with Eclipse.....	4
3.1 The source for our first OpenOffice application.....	9
4. Additional information.....	11

1. About

This short tutorial will show you the first steps of how to develop OpenOffice applications with Java using the Eclipse as development IDE. The whole tutorial assumes that you use a Windows version like Windows XP as operating system.

2. Setup your operating system

Before you can start with the development you have to prepare your operating system. This section will show which steps are necessary and in which order the setup will be done.

1. If not already done - download the latest OpenOffice version from the OpenOffice website. (<http://www.OpenOffice.org>). Install OpenOffice. Attention don't install OpenOffice into a path which includes white spaces! I will say you at point 5. why.
2. If not already done - download the latest version of Eclipse from the Eclipse website. (<http://eclipse.org/downloads/>) The latest version while writing this tutorial was 3.1. Parallel to this download you can download the latest you should download the latest Java Development Kit (JDK) from the Sun homepage (<http://java.sun.com/j2se/1.5.0/download.jsp>). The latest version of the JDK while writing this tutorial was JDK 5.0 Update 6. Also you should download the latest J2SE documentation. If all downloads finished you first have to install the JDK, after that you can unpack J2SE documentation to a folder of your choice. At least you can install Eclipse.
3. Download the latest OpenOffice SDK from <http://download.openoffice.org/680/sdk.html>. Unpack the SDK to a folder of your choice. e.g. D:\Download\OpenOffice\OpenOffice.org2.0_SDK. To use the OpenOffice SDK for Java development you have to need a few other third party tools. These tools are the windows port of GNU make which could be downloaded here <http://www.mingw.org/download.shtml> (Scroll down this page until you see the File List table inside this table search for the make tool. Install GNU make to a folder of your choice e.g. D:\Download\OpenOffice\make. The next tool you will need is Zip Tool 2.31 or higher. You can download this tool here <ftp://ftp.info-zip.org/pub/infozip/WIN32/> While writing this tutorial the latest version on this server was 2.31 (tool name on the server = zip231xN.zip) unpack the files to a folder of your choice e.g. D:\Download\OpenOffice\zipTool
4. Open a DOS command prompt window and change to the folder where you have unpacked the OpenOffice SDK. List the files. Among other files you will see configureWindows.bat and setsdkenv_windows.bat. You have to execute the configureWindows.bat batch file. It will ask you for your environments settings and at least writing all needed info to setsdkenv_windows.bat. The next questions will be asked:
 1. OpenOffice SDK path: The path where unpacked the SDK e.g. D:\Download\OpenOffice\OpenOffice.org2.0_SDK

2. The path to the OpenOffice installation path or to the URE (Uno Runtime Environment) installation path. We have no URE installed so we enter the OpenOffice.
 3. OpenOffice installation path. You can press enter without any input if your installation path is the same as shown in enter question between the [] other wise you have to enter your installation path.
 4. Path where you have GNU make installed. e.g.
D:\Download\OpenOffice\make\mingw\bin
 5. Path were unpack the zip tools e.g.
 6. You can skip the C++ compiler question if you don't have Visual Studio .net 2003 (VC++) installed or don't want to write C++ application that use OpenOffice. e.g.
 7. Same as 6., but for the .net framework and .net applications.
 8. Enter the Path to your JDK installation. If the path between [] is correct press enter otherwise enter your path.
 9. Enter a path for the SDK example output. Notice, if you use make to build the Java examples within the OpenOffice SDK the out put will be stored in this folder. Standard is the JDK path itself. You could direct the output e.g.
D:\Download\OpenOffice\OpenOffice.org2.0_SDK\JavaExampleOutPut (This path must exist!)
 10. Autodeployment: If you set this to yes compiled component examples will be automatically deployed to your OpenOffice installation. It's up to if you want this or not.
 11. The configuration is now finished and we **could** run setsdkenv_windows.bat, but have to do some more installations.
5. Before we can start we have to build the Java classes of the UNOIDL entities. Otherwise we couldn't use any of the possible types of the OpenOffice SDK. First you should create a new folder where the generated classes should be stored. e.g.
D:\Download\OpenOffice\OpenOffice.org2.0_SDK\OpenOfficeClasses. We build these classes using the javamaker application which is part of the OpenOffice SDK. You will find this tool in <your_OpenOffice_SDK_path>\windows\bin. The javamaker application is an command line tool so switch back to your Dos command window and change to the folder where your javamaker tool is located and run it with the following parameter.

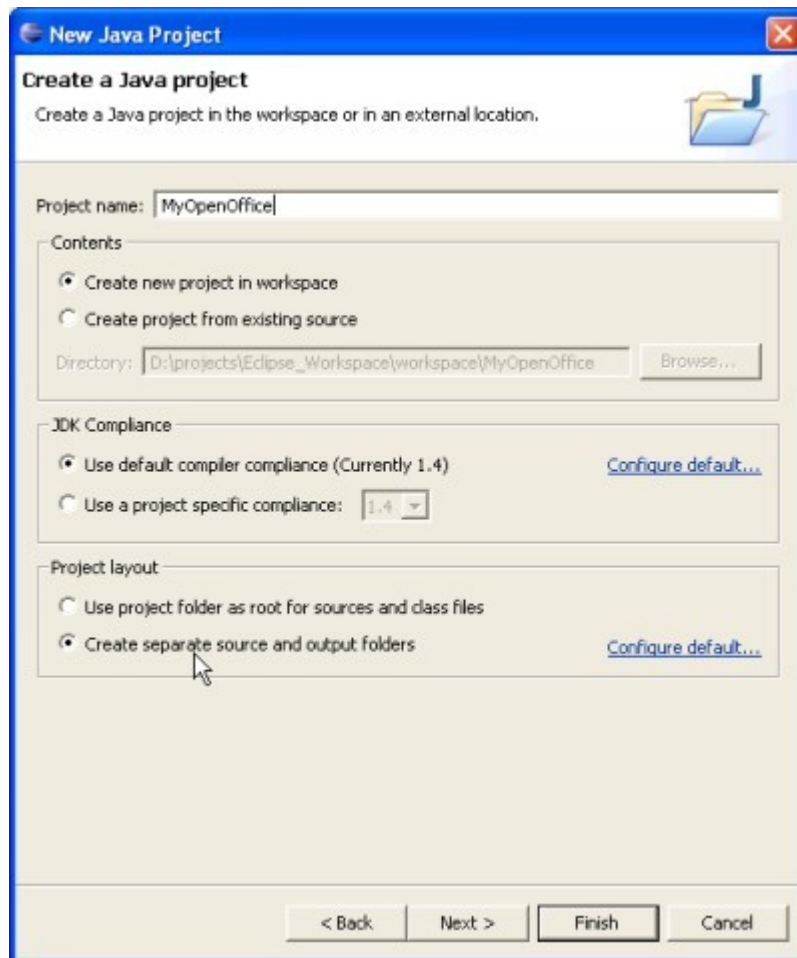
Javamaker -BUCR -O <oo_sdk_path>\OpenOfficeClasses <office_programm_dir>\types.rdb

Attention: javamaker doesn't like whitespace so that one of your both path contains whitespaces javamaker will throw errors like sal3.dll not found, init registries fail, .. If you have an english version of windows and installed OpenOffice in the standard path C:\Program Files\OpenOffice 2.0\ you will run into this trouble. If this happen you'd copy all dll's and rdb files to the same folder where javamaker is located and run javamaker again. Now, it javamaker should run without an error and you will find all classes in the target folder you specified. After all classes are build successfully you could delete the copied files. At least for C++ user that run into the same trouble using cppumaker it's the same workaround.

3. Our first OpenOffice Java application with Eclipse

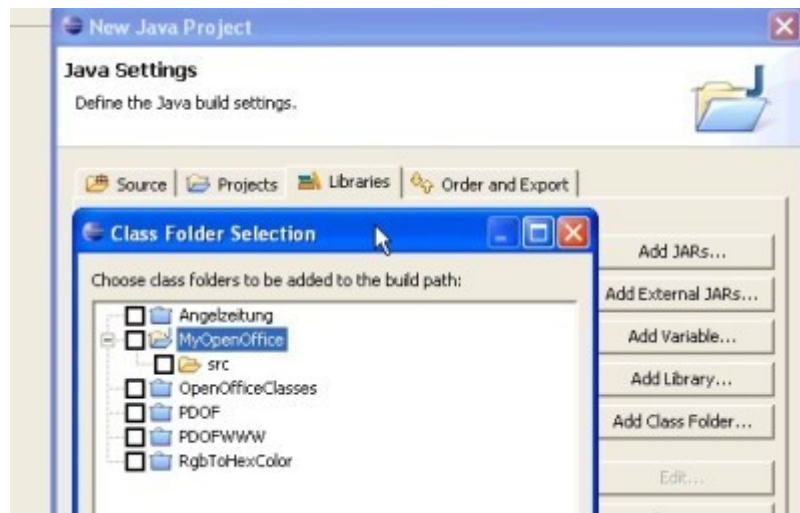
Now, you are ready to program your first OpenOffice Java application. Before we start switch back to your DOS command window change the OpenOffice SDK path and execute setsdkenv_windows.bat. This is necessary to set the environment variables.

Start Exclipse, now. If it's the first time you start Eclipse it will ask you to specify a workspace folder. This is the folder where your Eclipse projects will be stored. Choose a folder of your choice. The next step is creating a new Java project. You can do this via File->New->New Project. The project wizard window appear and you choose “Java Project” and click the Next button. In the following window you can set a project name and do some project settings. One of settings you should set is “Create separate source and output folders”. You could also set this as default. This step is shown in the following screenshot. Click “Next” to continue.



Picture 1: The Eclipse Project Wizard

In the next window you choose the tab page “Libraries” and then click “Add Class Folder”. The following window should appear. Use the “Create new Folder” button to create a new Folder. Make sure that your new Project is highlighted while doing that.



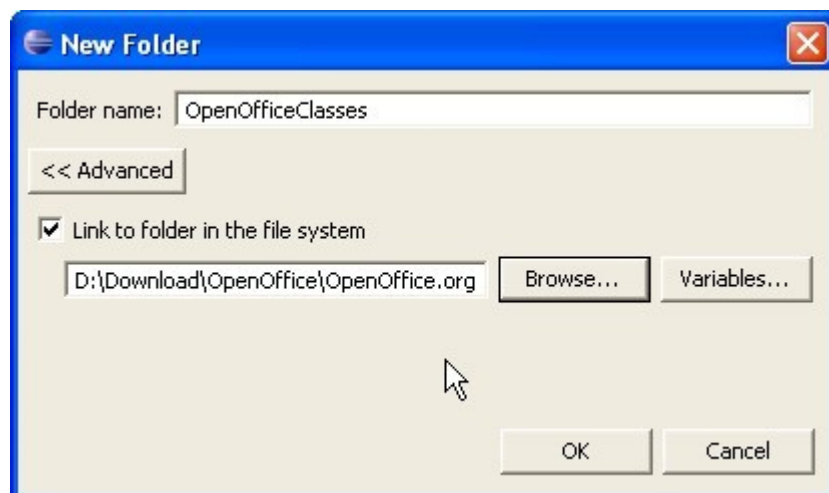
Picture 2: Create new Class Folder

Now the New Folder Dialog appear where you click the Advanced button.



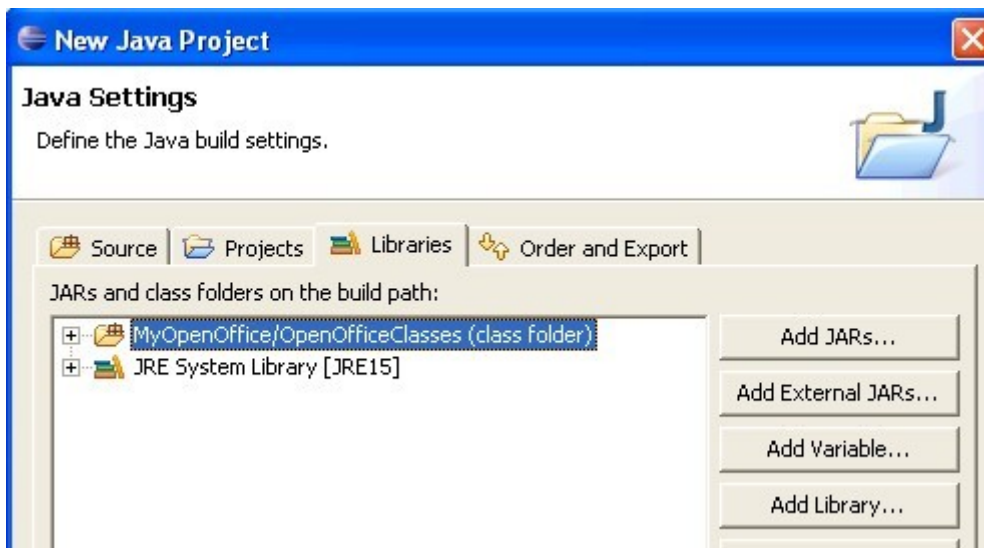
Picture 3: New Folder Dialog

After you clicked Advanced the New Folder Dialog will extend and you have to set the checkbox "Link to folder in the file system". Now, click Browse and choose the folder where have build the OpenOffice classes with javamaker. Finish this Dialog by clicking OK.



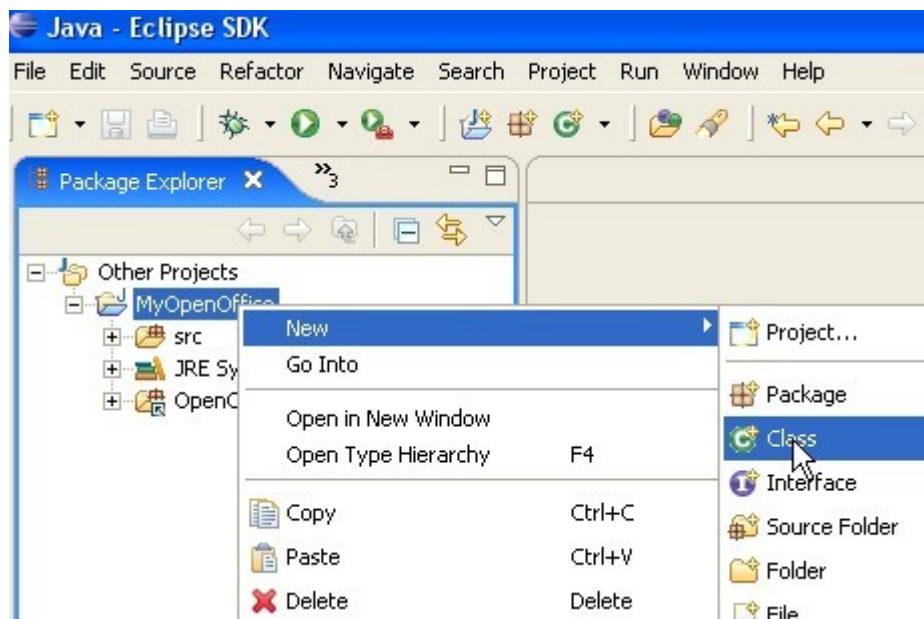
Picture 4: Choose your OpenOffice classes folder

Now, in your Libraries tab page should appear your linked Source Folder. Your tab page should now look like that.



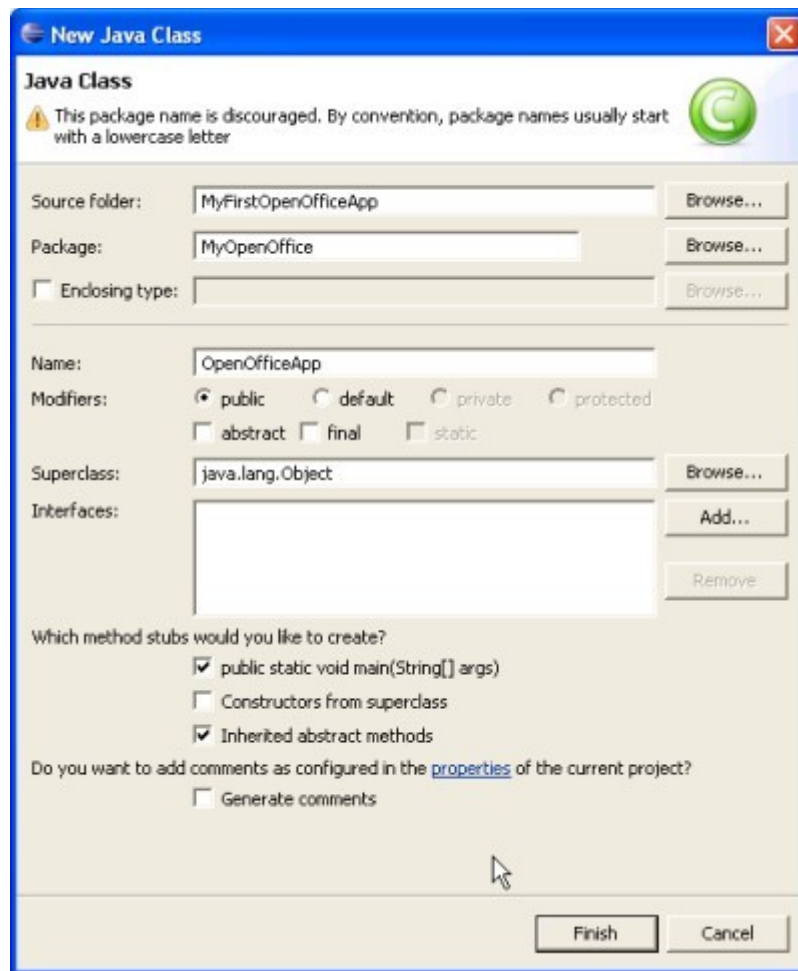
Picture 5: The Libraries tab page after add the Class Folder.

Now, your ready to add the first class to the project. The fastest way to do that is by right clicking on your project. This step is shown by the next picture.



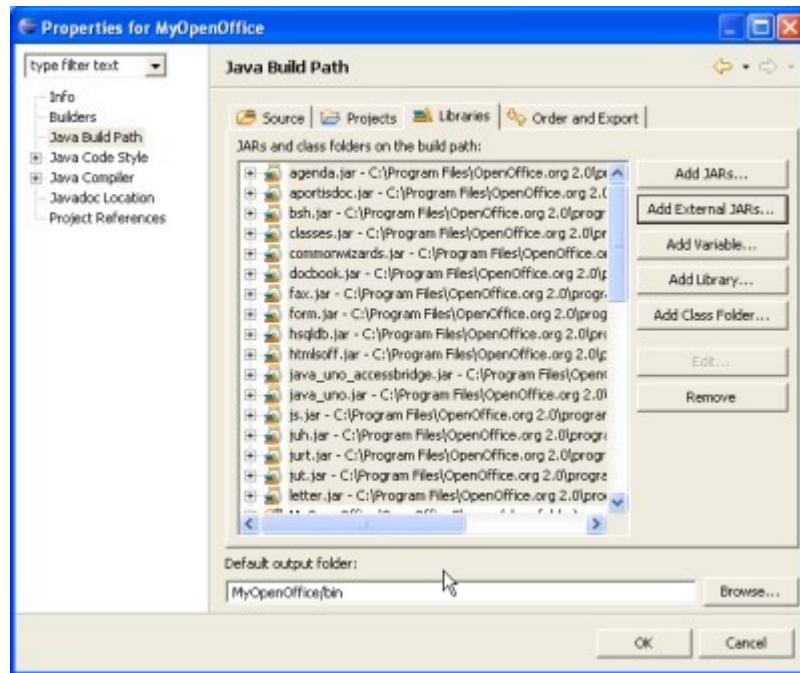
Picture 6: Add a new Class

Now, the “New Class” wizard should appear. Set MyOpenOffice as package name. The next step is choosing a name for new class. For our example we choose OpenOfficeApp, because this our only class we let us generate Eclipse also a main method. Therefor you need to select the checkbox “public static void main ..”. At least click finish. The following Picture will how you the last done steps.



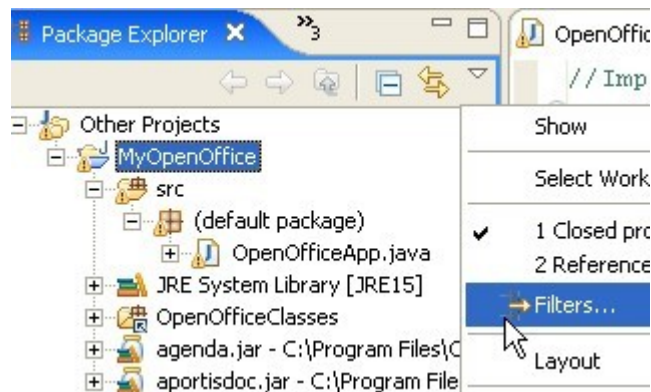
Picture 7: The new Class wizard.

The last step which is needed to complete the Project Settings is to add the existing OpenOffice Libraries to our new project. To add these libraries (Jars) right click the project name in the Package Explorer and choose “Properties” in the Project Properties Dialog choose “Java Build Path” and then choose the tab page Libraries. Click the Add External JARs button and navigate to your OpenOffice installation folder and look for the folder classes. In real you won't need all this libraries but to keep it simple you should select all and add them to your project. After that the window should look similar to the following screenshot.

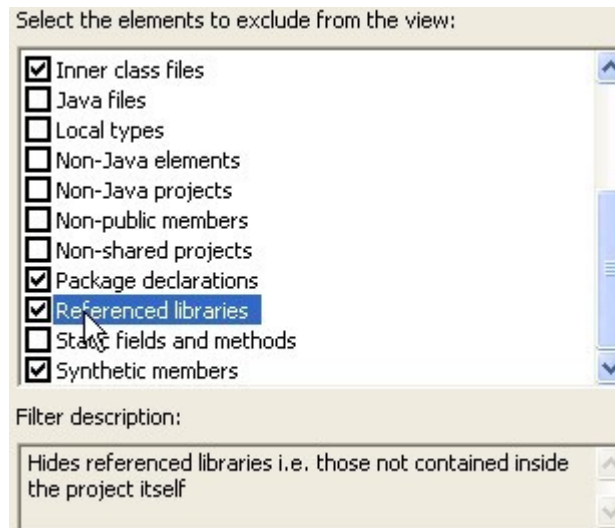


Picture 8: The new libraries

Really the last step with the project setting! Now, your Package Explorer looks a little bit overloaded all external packages are displayed. You could hide the external Jars by setting a Filter. The following screenshot will show how to do that. (use the reverse triangle)



Picture 9: The filter



Picture 10: Hide referenced libraries

3.1 The source for our first OpenOffice application

Similar to my C# tutorial I think show you the full sourcecode with many comments will be the best way. I think source code will explain more then only tell you all things step by step. But first, what will the application do? It will connect to an running instance of OpenOffice, create a new writer document, write some text into it and at least save it to the disk.

```

public class OpenOfficeApp
{
    public static void main(String[] args)
    {
        //Call the bootstrap to get the Component context
        com.sun.star.uno.XComponentContext oComponentContext = null;
        try
        {
            oComponentContext =
                com.sun.star.comp.helper.Bootstrap.bootstrap();
        }
        catch(com.sun.star.comp.helper.BootstrapException ex)
        {
            System.out.println(ex.getMessage());
        }

        if(oComponentContext != null)
        {
            try
            {
                //Get the service manager
                com.sun.star.lang.XMultiComponentFactory oMultiComponentFactory =
                    oComponentContext.getServiceManager();
                //Create a new desktop instance
                Object oDesktop =
                    oMultiComponentFactory.createInstanceWithContext(

```


4. Additional information

If you want to develop more applications using OpenOffice you will need further help. For further help you could use the [UNO online](#) help pages which are always up to date and the [UNO Developers Guide](#). All this material is also part of the SDK which you have downloaded.

Also you should visit the www.Oooforum.org in the [Snippet](#) and the [Macro an Api](#) forum you will find a lot of postings about OpenOffice development with Java.

To be continued ..

Lars Behrmann, lb@OpenDocument4all.com, 30. December 2005