XPontus XML Editor Developer Guide

About this document

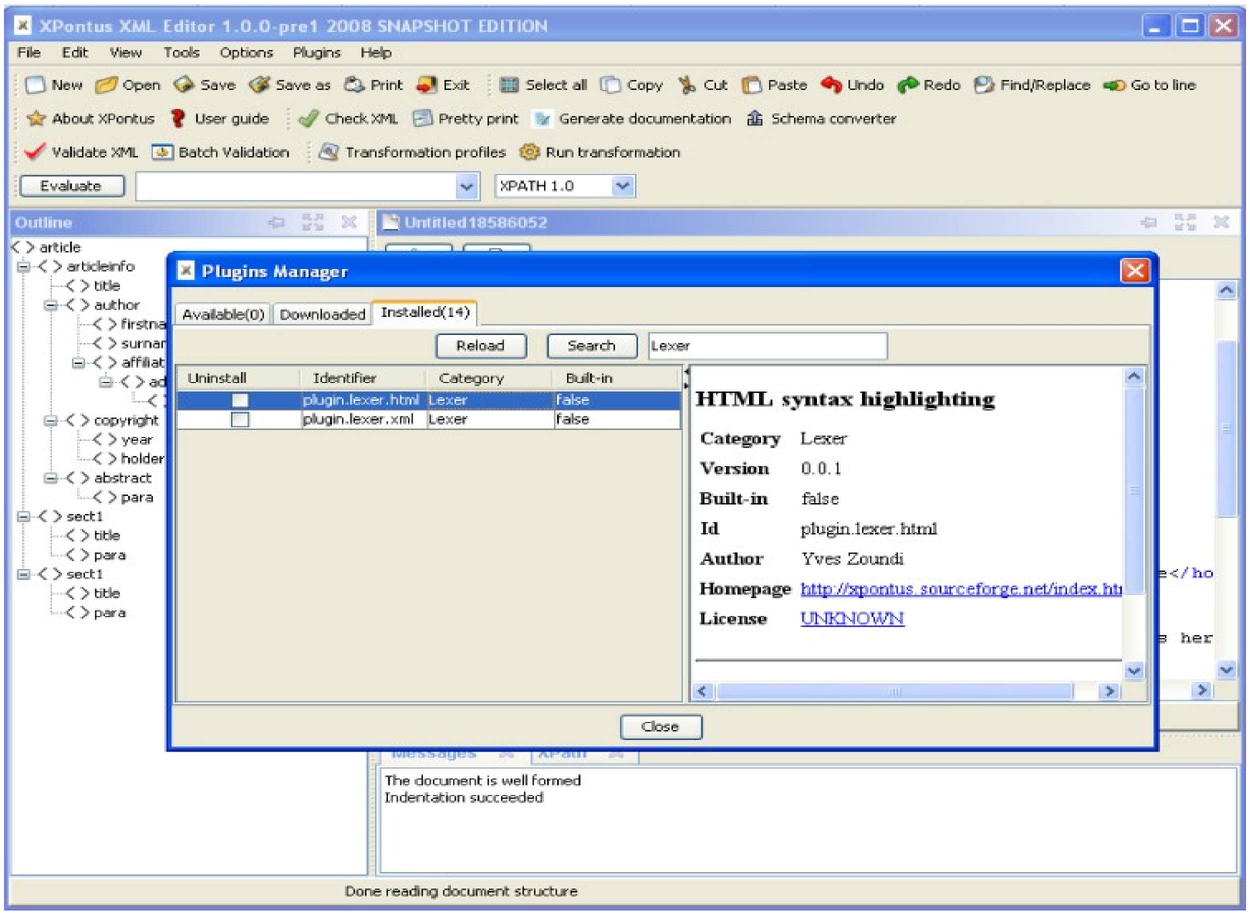
This document is a developer guide to XPontus XML Editor. It covers the main aspects of XPontus XML Editor. You'll hopefully learn how to the software works and how to build modules on XPontus XML Editor platform.

Intended audience

This document targets intermediary to experienced Java developers.

About XPontus XML Editor

XPontus XML Editor is a Java Swing based XML Editor. It's oriented towards text editing. It was a simple software but it's growing into a full blown platform. The software's homepage is : http://xpontus.sourceforge.net/index.html


XPontus XML Editor 1.0.0-pre1 2008 SNAPSHOT EDITION
File Edit View Tools Options Plugins Help
New Open Save Save as Print Exit Select all Copy Cut Paste Undo Redo Find/Replace Go to line
About XPontus User guide Check XML Pretty print Generate documentation Schema converter
Validate XML Batch Validation Transformation profiles Run transformation
Evaluate
XPATH 1.0
Outline
Untitled18586052
article
articleinfo
title
author
firstna
surnar
affiliat
copyright
year
holder
abstract
para
sect1
title
para
sect1
title
para
Plugins Manager
Available(0) Downloaded Installed(14)
Reload Search
Lexer
Close
Messages XPath

Uninstall	Identifier	Category	Built-in
☐	plugin.lexer.html	Lexer	false
☐	plugin.lexer.xml	Lexer	false

HTML syntax highlighting

Category	Lexer
Version	0.0.1
Built-in	false
Id	plugin.lexer.html
Author	Yves Zoundi
Homepage	http://xpontus.sourceforge.net/index.htı
License	UNKNOWN

```
The document is well formed
Indentation succeeded
```

Done reading document structure

Licence

Contents

Architecture

[TODO .... Some graphics and explation here]

Programming architecture

XPontus is not written using a pure MVC architecture. Most of the time here's how it happens :

- The model classes have a **java.beans.PropertyChangeSupport** object or they extend j*ava.util.Observable.*
- The model classes are bind to the model using JGoodies bindings or by implementing java.util.Observer interface in the view/controller class.
- The events binding : Most of the time I use reflection with the java.beans.EventHandler class

```
o.addActionListener(EventHandler.create(ActionListener.class, controller, "method"));

// the controller class would look like this

class SimpleController{

  void method(){}

}
```

Under the scenes

At the startup of the application :

- Default settings are created
- Plugins are collected and started
- The application window is created
- The event handlers of the visual plugins are added to the toolbar or the menubar or both
- The main window of the application is displayed

The main class of the application is *XPontusRunner* in the project ***xpontus_core***. It is located under the package *net.sf.xpontus.controllers.impl.*

Project code structure

The XPontus XML Editor maven project has 20 modules(sub-projects) :

- codecompletion_plugin : The code completion module
- evaluator_plugin : The experssion evaluator module
- gendoc_plugin : The documentation generator module
- i18n_plugin : The localization module
- icons_plugin : The icon pack module
- indentation_plugin : The code indentation module
- lexer_plugin : The text analysis module responsible for syntax coloring
- outline_plugin : The code structure plugin to display an outline of documents
- perspectives_plugin : The perspective module to present a document based on it's content type
- pluginsbrowser : The plugins manager module to add/remove/upgrade plugins
- preview_plugin : The transformation preview plugin to display the result of a transformation
- quicktoolbar_plugin : The quicktoolbar module which display some buttons for commons actions, based on the document type)
- scenario_plugin : The users transformation profiles module. Many engines such as Xalan, Saxon can be plugged to transform a document.
- schema_converter_plugin : The schema converter plugin which allows to generate schemas from document or to convert schemas to XML.
- scripting_plugin : The scripting module. You'll be able to develop Xpontus plugins using some scripting languages such as groovy, jruby, jython, beanshell, etc.
- themes_plugin : The themes pack module
- validation_plugin : The validation module which validates document
- vfs_plugin : Virtual file system module
- xmlsecurity : The XML signature module(sign and decrypt XML documents)
- **xpontus_core** : The heart of the application (you run Xpontus XML Editor from this project)

Requirements

Get the tools

- You need to install maven2(maven 2.0.8 is preferred). The maven bin folder must be in your PATH environment variable.
- Java (jdk 1.5 or later). Don't forget to target jdk 1.5 platforms if you're using Jdk 1.6 or later. I would like the software to keep running for users still using jdk 1.5
- An IDE like Eclipse, NetBeans, Jdeveloper, Idea or a simple text editor
- Access to the internet (or a Maven 2 repository)
- A subversion client like TortoiseSVN, SmartSVN, RapidSVN or the command line client

Get the sources

The latest version of the source code is in the trunk for now. To get it, use the Subversion client of your choice and checkout the folder :https://xpontus.svn.sourceforge.net/svnroot/xpontus/trunk/xpontus/

Import the project in your favorite IDE

If you are using a simple text editor. You can skip this step.

Eclipse users

In the xpontus directory run the command

```
mvn eclipse:eclipse
```

Idea users

In the xpontus directory run the command

```
mvn idea:idea
```

Netbeans users

You can open directly the project in Netbeans 5.x and later. I use Netbeans 6.0.

Building and running XPontus

Get the default settings archive

Please download the latest plugins.zip file from this address : http://xpontus.sourceforge.net/snapshot and extract it in your home folder. That will allow you to get the default settings and all the plugins.

Windows

Extract the archive in the folder *C:\documents and settings\myusername* where *myusername* is your login name. Overwrite any existing files or directories.

Unix/Linux

Extract the archive in the folder */home/myusername* where *myusername* is your login name. Please, overwrite any existing files or directories.

Building modules for XPontus

Introduction

Plugins are created using JPF architecture(http://jpf.sourceforge.net/). JPF is a simple plugin framework easier than OSGI(http://en.wikipedia.org/wiki/OSGi). OSGI is used by popular projects such as Eclipse(http://en.wikipedia.org/wiki/Eclipse_%28software%29). Every XPontus module requires the main module of the application called ***xpontus_core***. The ***xpontus_core*** module gives you access to the Xpontus XML Editor API.

Note : If you are developing an extension to an existing module the chances are that the module you're extending already imports the Xpontus XML Editor core API. (ie. You could create efficientValidation plugin in the Validation module.)

Create a new maven project

Create a maven project under the xpontus directory using maven command line or the IDE of your choice.

```
mvn archetype:create -DgroupId= com.mycompany.xpontus.plugins.mymodule -
   DartifactId= mymodule -Dpackagename= com.mycompany.xpontus.plugins.mymodule
```

Add required dependencies

The parent of your project will be xpontus and you should have the project **xpontus_core** as a dependency. Here is what your maven *pom.xml* should look like :

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
    <parent>
        <artifactId>xpontus</artifactId>
        <groupId>net.sf.xpontus</groupId>
        <version>1.0.0-pre1</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.mycompany.xpontus.plugins.mymodule</groupId>
    <artifactId>mymodule</artifactId>
    <name>MyModuleName</name>
    <version>1.0.0-pre1</version>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>net.sf.xpontus</groupId>
            <artifactId>xpontus_core</artifactId>
            <version>1.0.0-pre1</version>
            <scope>compile</scope>
        </dependency>
    </dependencies>
    <description>MyModule description</description>
    <build>
        <resources>
            <resource>
                <directory>src/main/java</directory>
                <excludes>
                    <exclude>*/**.java</exclude>
                    <exclude>*/**.form</exclude>
                </excludes>
            </resource>
        </resources>
        <plugins>
            <plugin>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <source>1.5</source>
                    <target>1.5</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

Create the plugin class

```
/*
 * My licence headers
 */
package com.mycompany.xpontus.plugins.mymodule;


import org.java.plugin.Plugin;


/**
 * Plugin description
 * @author Me
 */
public class MyModulePlugin extends Plugin {
    protected void doStart() throws Exception {
    }

    protected void doStop() throws Exception {
    }
}
```

Decide where to want to « plug » your plugin!

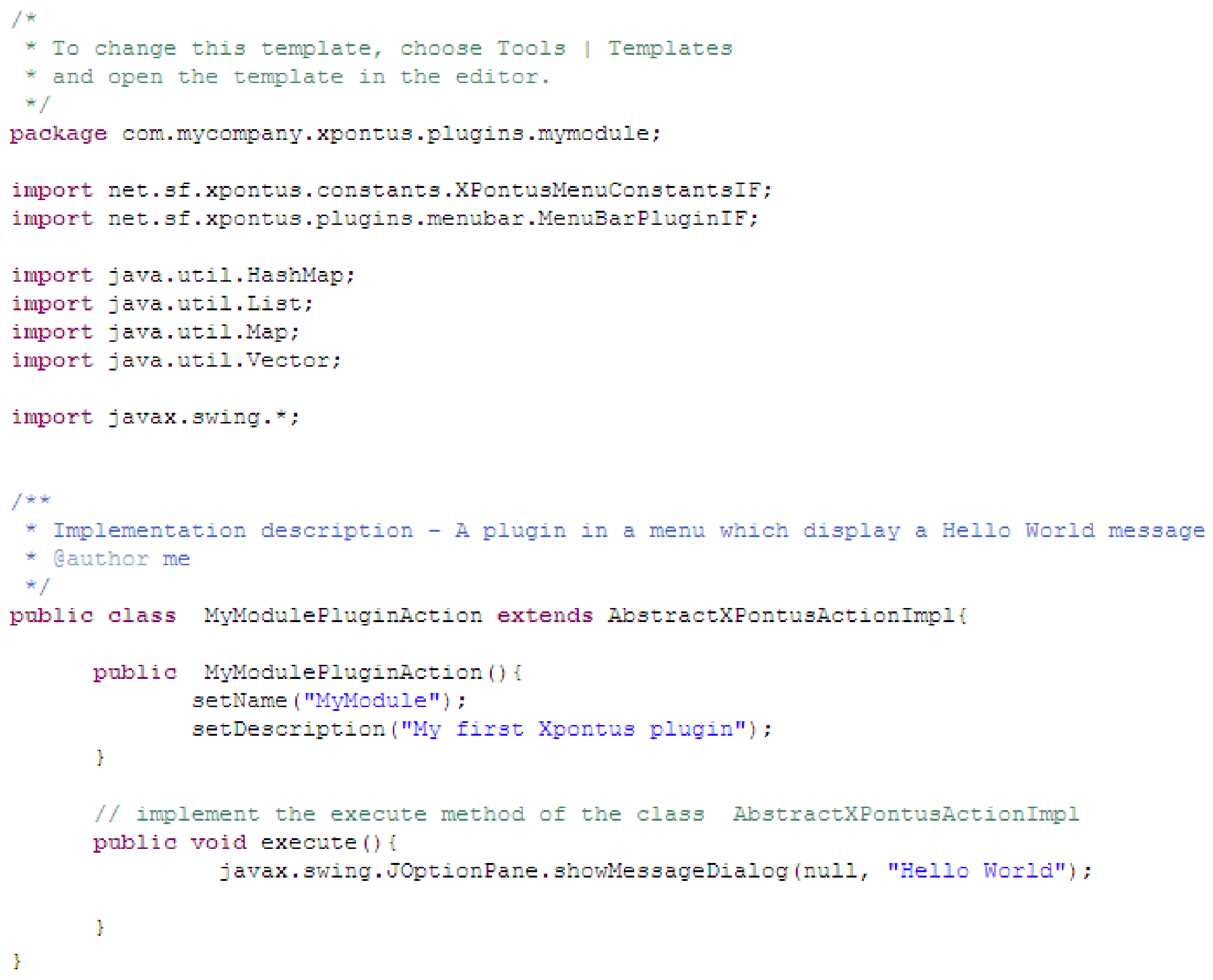
Do you want your plugin to appear in the menubar, the toolbar, both? Is your plugin a non visual plugin? Does your plugin integrate into an existing plugin?

Create the plugin interface implementation

Let's say we decide to add a plugin as a menu in the application's menubar. We'll implement the extension point interface of the menubar.

```java
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package com.mycompany.xpontus.plugins.mymodule;

import net.sf.xpontus.constants.XPontusMenuConstantsIF;
import net.sf.xpontus.plugins.menubar.MenuBarPluginIF;

import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Vector;

import javax.swing.Action;


/**
 * Implementation description - A plugin in a menu
 * @author me
 */
public class MyModulePluginImpl implements MenuBarPluginIF {
    private List<String> menuNamesList;
    private Map<String, List<Action>> actionsMap;
    private List<Action> actionsList;

    public List<String> getMenuNames() {
        if (menuNamesList == null) {
            menuNamesList = new Vector<String>();
            menuNamesList.add(XPontusMenuConstantsIF.FILE_MENU_ID);
        }

        return menuNamesList;
    }

    public Map<String, List<Action>> getActionMap() {
        if (actionsMap == null) {
            actionsMap = new HashMap<String, List<Action>>();
            actionsList = new Vector<Action>();
            actionsList.add(new MyModulePluginAction());
            actionsMap.put(getMenuNames().get(0), actionsList);
        }

        return actionsMap;
    }
}
```

Create the event which will be trigger from the menu of the plugin



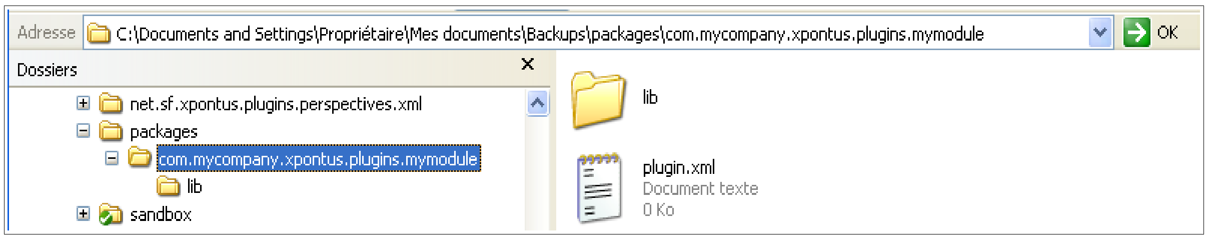
```java
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package com.mycompany.xpontus.plugins.mymodule;

import net.sf.xpontus.constants.XPontusMenuConstantsIF;
import net.sf.xpontus.plugins.menubar.MenuBarPluginIF;

import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Vector;

import javax.swing.*;


/**
 * Implementation description - A plugin in a menu which display a Hello World message
 * @author me
 */
public class  MyModulePluginAction extends AbstractXPontusActionImpl{

      public  MyModulePluginAction(){
             setName("MyModule");
             setDescription("My first Xpontus plugin");
      }

      // implement the execute method of the class  AbstractXPontusActionImpl
      public void execute(){
              javax.swing.JOptionPane.showMessageDialog(null, "Hello World");

      }
}
```

Create the plugin descriptor

```
<?xml version="1.0" ?>
<!DOCTYPE plugin PUBLIC "-//JPF//Java Plug-in Manifest 1.0"
"http://jpf.sourceforge.net/plugin_1_0.dtd">
<plugin id="com.mycompany.xpontus.plugins.mymodule" version="0.0.1" vendor="Me"
class="com.mycompany.xpontus.plugins.mymodule.MyModulePlugin">
    <attributes>
        <attribute id = "Built-in" value = "false"/>
        <attribute id="Category" value = "Tools"/>
        <attribute id="Homepage" value =
"http://www.mycompany.com/oss/xpontus/plugins/index.html"/>
        <attribute id="Description" value = "Test plugin"/>
        <attribute id = "DisplayName" value = "My first plugin"/>
        <attribute id = "License" value = "http://www.gnu.org/licenses/gpl-2.0.txt"/>
        <attribute id = "date" value = "26-02-2008"/>
    </attributes>
    <requires>
        <import plugin-id="plugin.core.menubar" />
    </requires>
    <runtime>
        <library id="com.mycompany.xpontus.plugins.mymodule.lib" path="lib/mymodule-1.0.0-
pre1.jar" type="code" >
            <export prefix="*"/>
        </library>
<!-- You can add dependencies this way, If your jar dependency already exists in the
xpontus_core module, you don't need to add it
        <library id="mydependency" version="1.0" path="lib/mydependency-1.0.jar"
type="code" >
            <export prefix="*"/>
        </library>   -->
    </runtime>

    <extension plugin-id="plugin.core.menubar" point-id="menubarpluginif"
id="mymoduleExtension">
        <parameter id="class"
value="com.mycompany.xpontus.plugins.mymodule.MyModulePluginImpl" />
    </extension>

</plugin>
```

Package the plugin

Here is the proposed directory structure to bundle your plugin


Adresse
C:\Documents and Settings\Propriétaire\Mes documents\Backups\packages\com.mycompany.xpontus.plugins.mymodule
OK
Dossiers
net.sf.xpontus.plugins.perspectives.xml
packages
com.mycompany.xpontus.plugins.mymodule
lib
sandbox
lib
plugin.xml
Document texte
0 Ko

Create a zip file containg the directory com.mycompany.xpontus.plugins.mymodule.

Install the plugin

- Launch Xpontus XML editor, open the plugins manager from the menu ***Plugins->Plugins Manager.*** Select the downloaded tab.
- Click on the ***Add plugin*** button and select your zip archive.
- Click on the ***Install*** button.

If you don't see any errors, that means that your plugin should be installed.

Test your plugin

Restart XPontus XML Editor and you should see your plugin under the menu File of the application.

Learning more about XPontus XML Editor

XPontus XML Editor news

You can read my blog at http://yveszoundi.blogspot.com

Mailing lists

You can find the mailing lists of the project here http://sourceforge.net/mail/?group_id=118253