Дата создания: 01.05.2007

Номинация: Офисные технологии

Название: Java + OpenOffice.org = генератор отчетов

Автор: Астафьев Артем Сергеевич

e-mail: ace@insurer.com.ua

Многие разработчики бизнес-приложений часто сталкиваются с необходимостью формировать в своей программе документы и печатать их. Эти документы условно можно разделить на две категории: «договора» и «отчеты». Обычно они создаются на основе некоторого шаблона. Для «договоров» этот шаблон представляет собой текст, общий для всех видов таких «договоров» (обычно его называют «рыба») с пустыми местами для данных, уникальных для каждого экземпляра «договора». Для «отчетов» шаблон обычно имеет табличный вид и в строках таблицы находятся однородные данные. Для формирования таких шаблонов и документов на основе этих шаблонов существует достаточное количество специальных программ, так называемых «генераторов отчетов». Они предоставляют удобный интерфейс для создания шаблонов и API для формирования и печати готовых документов на основе шаблонов.

В одном из проектов, над которым я работаю, возникла необходимость печатать как «отчеты», так и «договора». Сам проект написан на Java, поэтому и инструмент, с помощью которого можно было бы печатать документы должен как минимум иметь API для Java. Второе требование было, что инструмент должен быть как можно более гибким и открытым и желательно бесплатным. Мне пришлось столкнуться с работой с JasperReports 1.2.7 и в нем есть два огромных недостатка: если уже есть «рыба» документа в каком-то текстовом формате, то его нельзя непосредственно использовать для формирования шаблона и то,

что формирование шаблонов «договоров» в нем сильно затруднено. Здесь у меня возникла идея: а почему бы не использовать OpenOffice.org в качестве такого инструмента? Он обладает достаточной гибкостью, открытой архитектурой, очень развитым SDK для Java, может работать с широким спектром текстовых документов и он бесплатен! Можно написать небольшой код, который бы использовал существующий документ в качестве шаблона, вставлял туда нужные данные и печатал или сохранял его.

Для реализации этой идеи существует как минимум два подхода. В обоих подходах нужно сначала сформировать шаблон из существующей «рыбы» или создать с нуля. В этом шаблоне нужно неким образом пометить специальные места, куда наша программа будет вставлять данные. Второй этап – формирование конечного документа на основе шаблона и информации, специфичной именно для данного экземпляра документа. Так как формат документа Open Document открыт, хорошо документирован и основан на XML – можно написать свой обработчик данного формата, который бы в шаблон вставлял нужную информацию и формировал результирующий документ.

Я подробнее расскажу о втором подходе – загрузке шаблона и его обработке с помощью API OpenOffice.org.

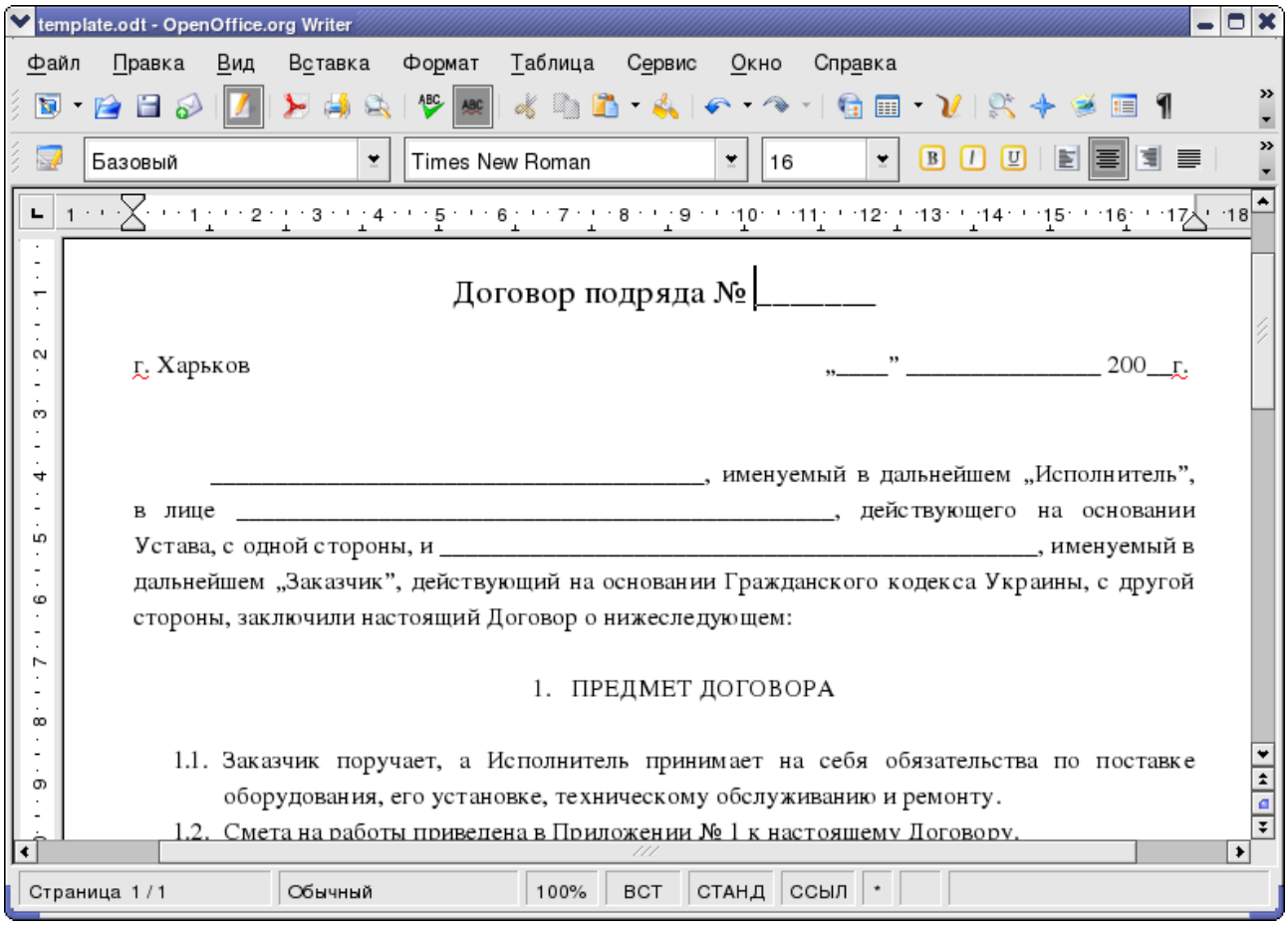
Для этого понадобится:

- jdk версии 1.5.0 или выше (http://java.sun.com/javase/downloads),
- OpenOffice.org SDK (http://download.openoffice.org/2.2.0/sdk.html),
- пакет OpenOffice.org версии 2.0 или выше.

По большому счету для рассматриваемой задачи можно и не использовать OpenOffice.org SDK, если в вашей сборке OpenOffice.org имеются нужные java-классы. Их можно найти в каталоге $OOO_HOME/program/classes, где $OOO_HOME – путь к установке OpenOffice.org. Если объем SDK для вас велик, а нужных файлов в

сборе нет, можно скачать на порядок меньший OpenOffice.org URE по адресу http://download.openoffice.org/2.2.0/ure.html, но в нем отсутствует документация разработчика. Нам понадобятся такие файлы: unoil.jar и juh.jar. Их нужно будет подключить к проекту в той среде, которую вы будете использовать для запуска этого примера. Или указать в classpath, если будете запускать из командной строки. Однако, для лучшего понимания приведенного кода советую как минимум прочитать введение в UNO в документации разработчика к OpenOffice.org SDK.

Возьмем какой-нибудь простую «рыбу» документа для примера. Теперь создадим простой, но наглядный шаблон с несколькими переменными. На рис.1 приведен исходный файл, на основе которого будем создавать шаблон.

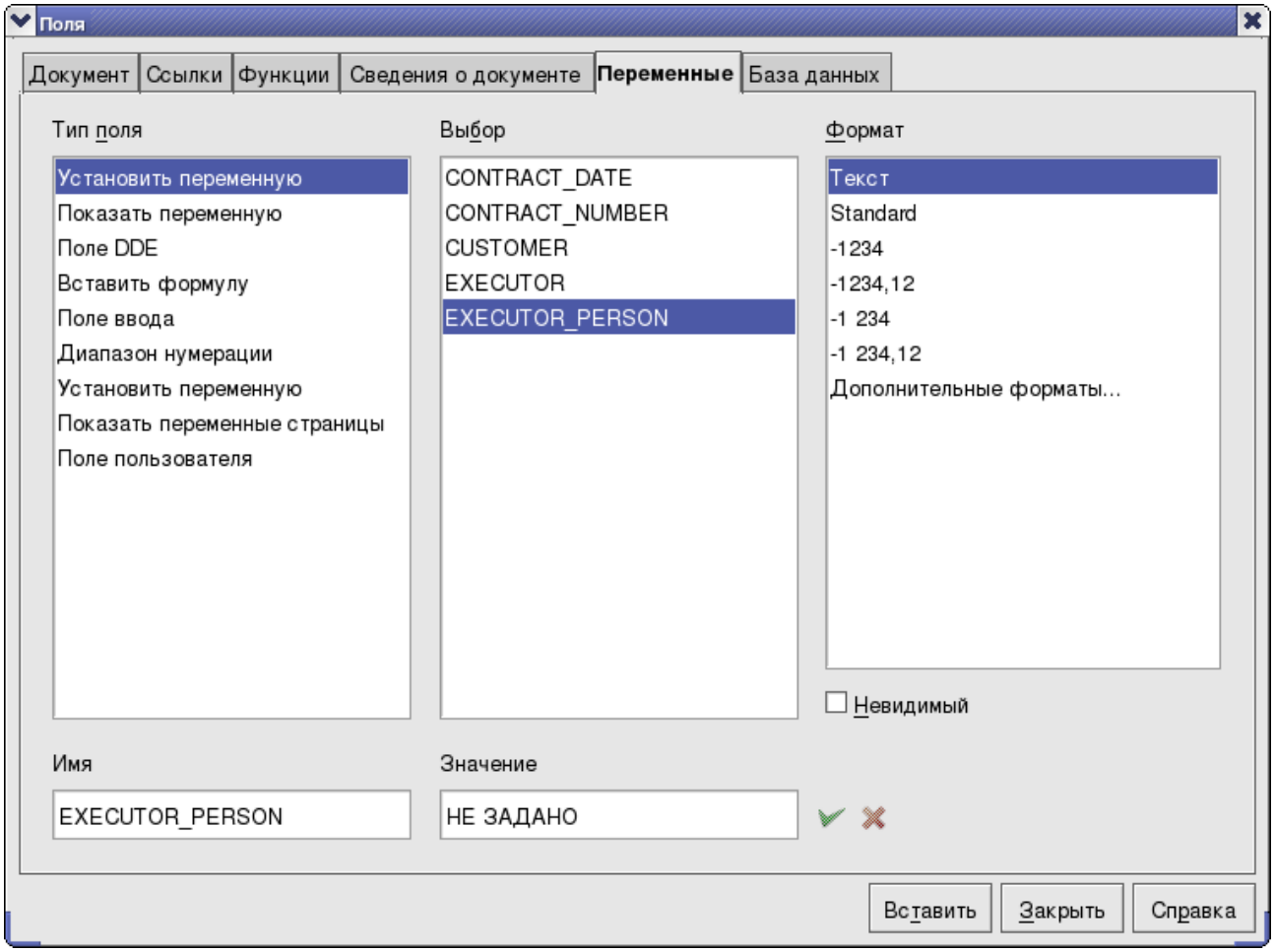


1: " "

Итак, первым этапом нужно в нашем шаблоне отметить места, в которые потом наша программа будет вставлять нужную информацию. Сделать это можно несколькими способами, я остановлюсь на наиболее простом. Будем использовать так называемые «поля», в частности одну из их разновидностей – переменные. Идея заключается в том, чтобы вставлять в нужные места шаблона переменные с разными именами, а наша программа потом будет в документе заменять значения этих переменных на нужные.

Для того, чтобы вставить переменную, установите курсор на нужное место в тексте, выберите в меню Вставка->Поля->Дополнительно или нажмите Ctrl+F2. В появившемся диалоге выберите вкладку «Переменные», в списке «Тип поля» выберите «Установить переменную», а в списке «Формат» выберите «Текст».

Далее осталось самое главное – в поле ввода «Имя» ввести имя переменной, а в поле «Значение» - ее первоначальное значение. В



2:

нашей задаче значение может быть любым и для наглядности условимся, что значение у всех переменных будет «НЕ ЗАДАНО». Имя переменной должно быть уникальным в рамках данного шаблона, но если нужно одну и ту же переменную использовать несколько раз, то можно сначала установить значение переменной как описано выше, а потом использовать поле другого типа: «Показать переменную» и указать только имя переменной. На рис. 2 приведен диалог ввода переменных. Имена переменных соответствуют следующим полям: CONTRACT_DATE – дата заключения договора, CONTRACT_NUMBER – номер договора, CUSTOMER – Заказчик, EXECUTOR – Исполнитель, EXECUTOR_PERSON – представитель Исполнителя. Сохраним полученный файл под именем template.odt. Вот, что у нас в результате получилось:

Договор подряда № НЕ ЗАДАНО

г. Харьков НЕ ЗАДАНО

НЕ ЗАДАНО, именуемый в дальнейшем „Исполнитель”, в лице НЕ ЗАДАНО, действующего на основании Устава, с одной стороны, и НЕ ЗАДАНО, именуемый в дальнейшем „Заказчик”, действующий на основании Гражданского кодекса Украины, с другой стороны, заключили настоящий Договор о нижеследующем:

1. ПРЕДМЕТ ДОГОВОРА

1.1. Заказчик поручает, а Исполнитель принимает на себя обязательства по поставке оборудования, его установке, техническому обслуживанию и ремонту.

2. Смета на работы приведена в Приложении № 1 к настоящему Договору.

Кстати, этот документ тоже можно использовать в качестве шаблона. Можете убрать текст статьи и оставить только текст договора, а можете ничего не менять.

Наконец, переходим ко второму этапу – обработке шаблона с помощью нашей программы. Текст программы приведен в Листинге 1. Я постарался прокомментировать места, которые могут показаться непонятными, а сам принцип работы вкратце следующий. Как параметр программе передается имя файла с шаблоном, а значения переменных для простоты прописаны в самом коде, хотя их, например, тоже можно загружать из какого-нибудь текстового файла. В HashMap variableMap хранятся соответствия имен переменных их значениям и в начале программы он для наглядности вручную заполняется. Далее вызывается функция connect(), которая устанавливает соединение с OpenOffice.org и устанавливает внутренние переменные, которые в дальнейшем используются в разных местах кода. Потом с помощью функции openDocument открывается документ и получается его интерфейс XtextFieldsSupplier. Он в свою очередь имеет функцию getTextFields, которая возвращает перечисление xTextFieldsEnumeration всех «полей» в документе. Теперь осталось пройтись по всем текстовым полям, отобрать из них нужные и установить их значения в соответствии variableMap.

В цикле для каждого поля проверяется, поддерживает ли он сервис com.sun.star.text.TextField.SetExpression – «установить переменную». Если это так – значит это наше специальное поле и для него запрашивается набор свойств xPropertySet. Значение свойства VariableName как раз и является именем нашей переменной, которое мы

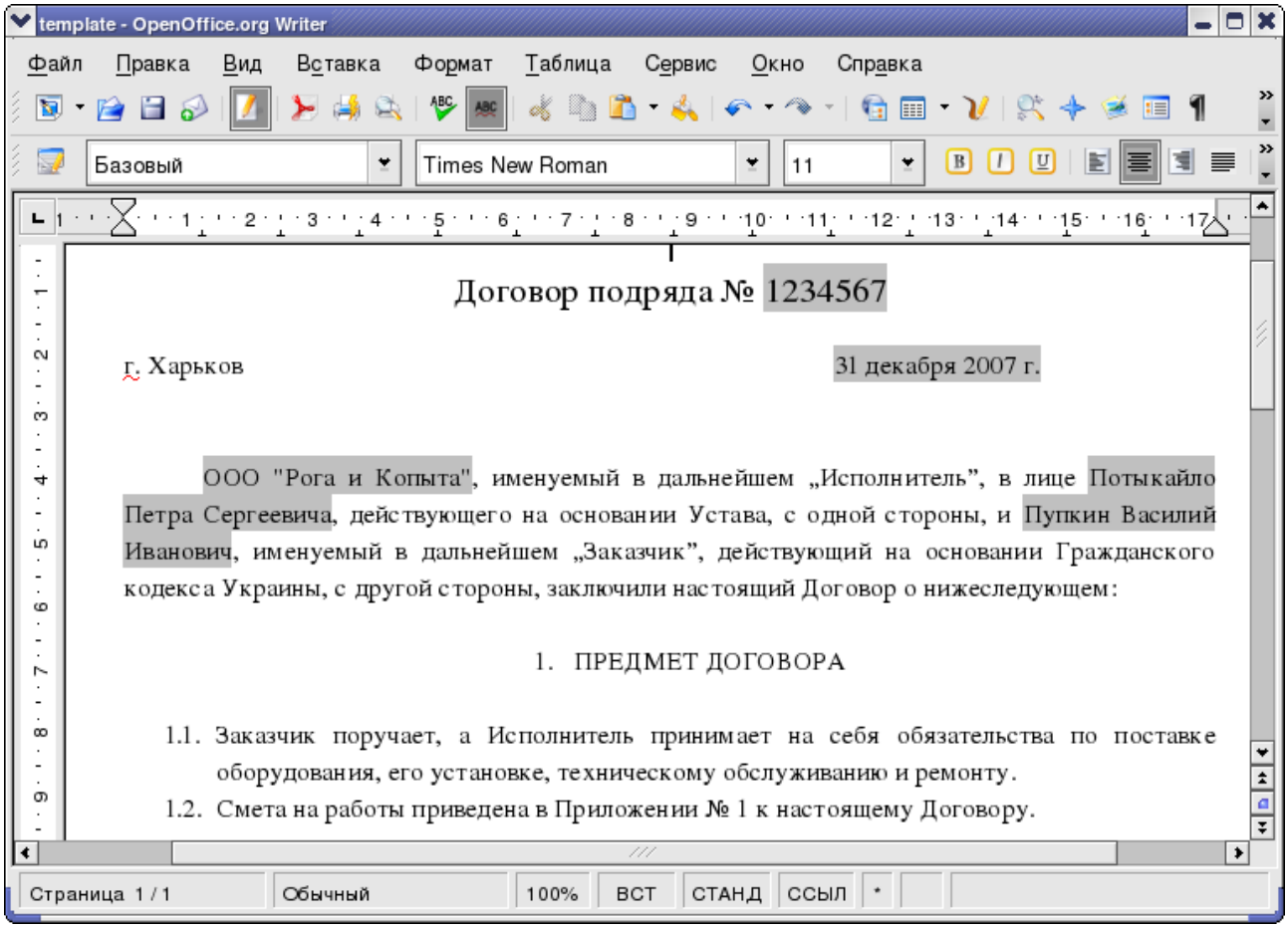
установили в шаблоне. Затем по имени переменной в variableMap ищется ее значение и устанавливается как свойство поля Content. Чтобы не было проблем с отображением значений устанавливается еще два свойства поля – IsVisible для маркировки поля как видимого и SubType для установки типа поля как строкового. Хотя выше при установке поля я и писал, что нужно устанавливать тип поля как «Текст» и не говорил, чтобы устанавливали свойство «Невидимое», но для перестраховки все-таки в коде программы явно устанавливаю эти свойства. Далее нужно обновить отображаемые значения переменных на экране, это делается с помощью функции refresh() интерфейса XRefreshable.

Все что нам осталось сделать – сохранить или распечатать полученный документ. Сохранить можно с помощью функции saveDocument или saveAsDocument. Первая сохраняет документ под текущим именем, что в нашем примере не понадобится, так как будет испорчен шаблон. Вторая сохраняет документ под новым именем – чтобы, например, передать его по электронной почте. Распечатать документ можно тоже двумя способами: вызвать функцию printDocument и это будет равносильно нажатию кнопки печати на принтере по умолчанию или вызвать функцию executeCommands. Эта функция использует специальный механизм обработчиков команд OpenOffice.org. Как параметр ей передается массив строк – имен команд, которые нужно выполнить. Имена команд являются обозначениями тех действий, которые может выполнить OpenOffice.org. Например, команда Print равносильна выбору в меню Файл->Печать, Save – Файл->Сохранить и т.д. В данном примере используется команда Print, которая выведет диалог печати и даст пользователю возможность произвести различные настройки печати.

Эта команда выполняется асинхронно, поэтому есть возможность, что документ еще не будет подготовлен для печати, а мы уже будем

пытаться его закрыть – в таком случае просто ничего не распечатается. Здесь приходится применить хитрость – пытаться закрыть документ до тех пор, пока будет происходить исключение com.sun.star.util.CloseVetoException – знак того, что документ еще не подготовлен для печати.

При запуске программы не забудьте указать в качестве параметра имя файла-шаблона и указать в classpath пути к библиотекам URE. В результате запуска получаем готовый документ, приведенный на рис. 3.



3:

В результате на напечатанном документе серого фона не будет видно, это просто специально сделано на экранной форме для маркировки мест, в которых находятся «поля».

Заключение

И пусть не ругают меня гуру Java за статические переменные, отсутствие обработки некоторых исключений, отсутствие проверки возвращаемых значений на null и т.п. Данный пример был сделан на основе небольшой части разрабатываемого мной большого проекта, и код пришлось сильно сокращать, чтобы поместить в рамки статьи. Тем более, что задача данной статьи – знакомство с возможностями API OpenOffice.org на реальном примере и указаний возможных вариантов решения тех или иных проблем, а не выдача готового «красивого» решения.

В любом случае это решение успешно работает в реальном проекте. Конечно кода в нем на порядок больше но и возможностей соответственно тоже больше.

Есть еще огромное количество возможностей, которые можно реализовать. Например, можно выгружать нужные данные в Calc, на основании которых делать различные диаграммы. Или сделать отчеты табличного вида с помощью Calc. Можно расширить возможности данного примера: сделать загрузку файлов из памяти (например, хранимых в БД в BLOB) или сделать специальный редактор шаблонов, который бы позволял удобно расставлять переменные и как-то автоматически связывать их с данными из вашей программы. Специалисты скажут: но ведь есть большое количество различных способов передать в документы данные из внешних источников, например, с помощью «источников данных». Но каждый способ оптимален для каждого конкретного случая, и в моем случае описанный в данной статье вариант был единственно возможным. Может для кого-то он тоже окажется полезным.

```
package org;

import java.util.HashMap;
import javax.swing.*;
import java.io.*;

import com.sun.star.beans.*;
import com.sun.star.util.*;
import com.sun.star.io.*;
import com.sun.star.connection.*;
import com.sun.star.lang.*;
import com.sun.star.text.*;
import com.sun.star.frame.*;
import com.sun.star.container.*;

import com.sun.star.view.XPrintable;
import com.sun.star.uno.UnoRuntime;
```

```
import com.sun.star.uno.XInterface;
import com.sun.star.uno.XNamingService;
import com.sun.star.uno.XComponentContext;

public class OOOReportGenerator {

  private static XComponentContext xRemoteContext = null;
  private static XMultiComponentFactory xRemoteServiceManager = null;
  private static XURLTransformer xTransformer = null;
  private static XComponentLoader xComponentLoader = null;
  private static XDesktop xDesktop = null;

  public OOOReportGenerator() {
  }

  /*                          OpenOffice.org
   *
   */
  public static void connect() throws Exception {
```

```
//
xRemoteContext = com.sun.star.comp.helper.Bootstrap.bootstrap();
xRemoteServiceManager = xRemoteContext.getServiceManager();

//                ,
Object transformer = xRemoteServiceManager.createInstanceWithContext(
    "com.sun.star.util.URLTransformer", xRemoteContext );
xTransformer = (XURLTransformer)UnoRuntime.queryInterface(
    XURLTransformer.class, transformer);

//                Desktop
Object desktop = (XInterface) xRemoteServiceManager.createInstanceWithContext(
    "com.sun.star.frame.Desktop", xRemoteContext);
xDesktop = (XDesktop)UnoRuntime.queryInterface(
    XDesktop.class, desktop);

//
xComponentLoader = (XComponentLoader)UnoRuntime.queryInterface(
    XComponentLoader.class, desktop);
```

```
}

/*                  ,                        sURL
 *                          ,      sURL -
 */
public static XComponent openDocument(String sURL) throws Exception {
  //                              URL
  java.io.File sourceFile = new java.io.File(sURL);
  StringBuffer sTmp = new StringBuffer("file:///");
  sTmp.append(sourceFile.getCanonicalPath().replace('\\', '/'));
  sURL = sTmp.toString();

  PropertyValue[] loadProps = new PropertyValue[0];
  return xComponentLoader.loadComponentFromURL(sURL, "_blank", 0, loadProps);
}

/*                  ,       askIfVetoed=true,
 *
 */
```

```
  public static boolean closeDocument(XComponent comp, boolean askIfVetoed) {
    XCloseable c = (XCloseable)UnoRuntime.queryInterface(
        XCloseable.class, comp);
    boolean dispose = true;
    try {
      c.close(false);
    } catch (com.sun.star.util.CloseVetoException e) {
      if ( askIfVetoed ) {
        int action = JOptionPane.showConfirmDialog(null, "
!" +
            "                              ?",
            "       ", JOptionPane.WARNING_MESSAGE, JOptionPane.YES_NO_OPTION);
        if ( JOptionPane.NO_OPTION == action ) {
          dispose = false;
        }
      }
      if ( dispose ) {
        comp.dispose();
      }
```

```
  }
  return dispose;
}

/*
 */
public static void printDocument(XComponent comp)
throws com.sun.star.lang.IllegalArgumentException {
  XPrintable xPrintable = (XPrintable)UnoRuntime.queryInterface(
      XPrintable.class, comp);
  PropertyValue[] printOpts = new PropertyValue[0];
  xPrintable.print(printOpts);
}

/*
 */
public static void saveDocument(XComponent comp, PropertyValue[] props) {
  XStorable store = (XStorable)UnoRuntime.queryInterface(
      XStorable.class, comp);
```

```
    saveAsDocument(comp, store.getLocation(), props);
  }

  /*                                ,                        aURL
   */
  public static void saveAsDocument(XComponent comp, String aURL, PropertyValue[]
props) {
    XStorable store = (XStorable)UnoRuntime.queryInterface(
        XStorable.class, comp);
    try {
      store.storeToURL(aURL, props);
    } catch (Exception e) {
      System.out.println( "                        !" + e );
    }
  }

  /*
   *
   *
```

```
   */
  public static void executeCommands( String[] commands )
  throws com.sun.star.uno.Exception {
    //
    XFrame xFrame = xDesktop.getCurrentFrame();
    //                          DispatchProvider.
    XDispatchProvider xDispatchProvider =
(XDispatchProvider)UnoRuntime.queryInterface(
        XDispatchProvider.class, xFrame );
    for ( int n = 0; n < commands.length; n++ ) {
      //            URL
      com.sun.star.util.URL[] aURL  = new com.sun.star.util.URL[1];
      aURL[0] = new com.sun.star.util.URL();
      com.sun.star.frame.XDispatch xDispatch = null;

      aURL[0].Complete = ".uno:" + commands[n];
      xTransformer.parseSmart( aURL, ".uno:" );

      //                                                  URL
```

```
      xDispatch = xDispatchProvider.queryDispatch( aURL[0], "", 0 );
      if ( xDispatch != null ) {
        com.sun.star.beans.PropertyValue[] lParams = new
com.sun.star.beans.PropertyValue[0];
        Object obj = xRemoteServiceManager.createInstanceWithContext(
            "com.sun.star.frame.DispatchHelper", xRemoteContext );
        XDispatchHelper dh = (XDispatchHelper)UnoRuntime.queryInterface(
            XDispatchHelper.class, obj);
        dh.executeDispatch(xDispatchProvider, aURL[0].Complete, "", 0, lParams);
      } else {
        System.out.println( "                          " + aURL[0].Complete );
      }
    }
  }

  public static void main(String[] args) {
    HashMap<String,String> variableMap = new HashMap<String,String>();

    variableMap.put("CONTRACT_NUMBER", "1234567");
```

```
    variableMap.put("CONTRACT_DATE", "31         2007  .");
    variableMap.put("EXECUTOR", "    \"            \"");
    variableMap.put("EXECUTOR_PERSON", "                        ");
    variableMap.put("CUSTOMER", "                      ");

    try {
      connect();

      XComponent currentDocument = openDocument(args[0]);

      XTextFieldsSupplier xTextFieldsSupplier =
(XTextFieldsSupplier)UnoRuntime.queryInterface(
          XTextFieldsSupplier.class, currentDocument);

      //
      XEnumerationAccess xEnumerationAccess = xTextFieldsSupplier.getTextFields();
      XEnumeration xTextFieldsEnumeration = xEnumerationAccess.createEnumeration();
      XRefreshable xRefreshable = (XRefreshable)UnoRuntime.queryInterface(
          XRefreshable.class, xEnumerationAccess);
```

```
      while ( xTextFieldsEnumeration.hasMoreElements() ) {
        Object service = xTextFieldsEnumeration.nextElement();
        XServiceInfo xServiceInfo = (XServiceInfo)UnoRuntime.queryInterface(
            XServiceInfo.class, service);

        if (xServiceInfo.supportsService("com.sun.star.text.TextField.SetExpression"))
{
          XPropertySet xPropertySet = (XPropertySet)UnoRuntime.queryInterface(
              XPropertySet.class, service);
          String name = (String)xPropertySet.getPropertyValue("VariableName");
          Object content = variableMap.get(name);
          xPropertySet.setPropertyValue("SubType",
              new Short(com.sun.star.text.SetVariableType.STRING));
          xPropertySet.setPropertyValue("Content",
              content == null ? " " : content.toString());
          xPropertySet.setPropertyValue("IsVisible", true);
        }
      }
```

```
xRefreshable.refresh();

String[] cmds = {"Print"};
executeCommands(cmds);
/*
 * printDocument(currentDocument) -
 * saveDocument (currentDocument, props) -
 * saveAsDocument (currentDocument, sURL, props) -
 */

closeDocument(currentDocument, false);
XCloseable c = (XCloseable)UnoRuntime.queryInterface(
    XCloseable.class, currentDocument);
while(true) {
  try {
    c.close(false);
    break;
  } catch (com.sun.star.util.CloseVetoException e) {
  }
```

```
        Thread.sleep(200);
      }
    } catch(Exception e) {
      System.out.println("                          ");
    }
  }
}
```