

Дата создания: 9.05.2007

Номинация: «ОФИСНЫЕ ТЕХНОЛОГИИ»

Название: Java решение OpenOffice файл сохранить как ==> HTML.

Автор: Феофилов Иван Игоревич

e-mail: fivig@mail.ru

В статье описывается проект основанный непосредственно на OpenOffice, обсуждаются примеры работы с офисом через Java API. В первую очередь предназначена тем кто хочет научиться работать с OpenOffice на Java, но не знает с чего начать. Так же будет полезна для ознакомления с возможностями API OpenOffice.

Конвертор презентаций разрабатывался, как дополнение к внедряемой в университете системе «Дистанционного обучения», для обеспечения возможности просмотра презентаций стандартным браузером по http. Преподаватели создают презентации занятий в MS PowerPoint или OpenOffice, разрабатываемое приложение должно обеспечивать экспорт презентации из любого формата который может открыть OpenOffice в HTML. Основными требованиями были кроссплатформенность приложения планируется эксплуатация в виде Web Services , возможность изменения качества экспортируемых изображений, исключение использования разных файлов для одинаковых изображений и в частности для фона.

Началось все с того что, преподаватель рассказал о неудачном проекте, решения описанной задачи средствами C#, Microsoft Office, VB. Решение оказался неудачным в том плане, что работало только на платформе Windows в связке с .Net, в оконном режиме и имело ряд критичных недоработок. Перекапывать чужой, недокументированный код, как наверное большинство других программистов терпеть не могу, да и проект был небольшой. Поэтому решил переписать практически с нуля.

Так получилось, что OpenOffice я в глаза не видел, но в решаемой задаче ему отводилось центральное звено, Поэтому я начал знакомство с базовой функциональностью. Первое, что пришло в голову это использовать встроенную функцию сохранить как, но после пары экспериментов понял что там пока ситуация безнадежная в плане грубой реализации. Функция сохранить в HTML в Ooo 2.0 как и в MS Office 2003 (2007 уже умеет) реализована простым экспортом каждой страницы в в jpeg изображение, которые объединяются в набор ссылающихся друг на друга html страниц. Вариант не подходит из- за низкого качества изображения и большого размера файлов. Есть еще возможность экспорта в Ooo которая работает не корректно т.е фигуры вставленные в презентацию с помощью панели инструментов рисунок не экспортируются совсем. Связано это с тем что для экспорта используются таблицы стилей корневая лежит здесь <OpenOffice_HOME>\share\xslt\export\xhtml\ooo2xhtml.xsl . А технология xslt не может физически преобразовывать фигуры хранящиеся в формате XML презентации в растровые изображения.

Осталось выбрать вариант решения: дорабатывать существующий xslt(примерно 3000 строчек преобразований), для решения таким способом нужно как минимум прекрасно разобраться с внутренним форматом файлов либо OpenOffice и добавить функцию экспорта фигур. Вариант честно говоря просто не понравился тем что пришлось бы изучить слишком много информации о внутреннем формате OpenOffice, польза которой в дальнейшем весьма сомнительна. Остается вариант написать все на Java обращаясь к объектам презентации на Java через UNO с небольшим вкраплением xslt.

За месяц удалось выкроить несколько дней чтобы разобраться какие же все

таки технологии нужны для успешной реализации, и сможет ли OpenOffice предоставить требуемую функциональность, оказалось на 90% да! В результате получился следующий план действий:

- Экспорт объектов (фигур и текста)в виде изображений в формате EMF с поддержка фонов слайдов.
- Экспорт информации о положении и параметрах всех объектов в XML файл описания объектов.
- Экспорт названий слайдов
- XSLT преобразование XML файла описывающего презентацию в набор HTML страниц
- Преобразование EMF изображений в PNG (векторные объекты, объекты с прозрачностью) и JPEG (растровые изображения без поддержки прозрачности, фон) изображения.

Как в любом новом деле я начал копать необходимые для разработки ресурсы. Первым и пожалуй самым полезным оказался Ooo-SDK_2.0. Или по-русски пакет OpenOffice, для разработки новых компонентов. Включающий в себя набор из множества примеров исходного кода, библиотек, бинарных файлов, описаний IDL файлов которые имеют стабильный API и могут быть расширены для получение необходимой функциональности. Тут я естественно ответы на все возникающие вопросы не нашел и отправился за помощью к google. Он вывел на основной форум пользователей и разработчиков, любителей OpenOffice www.ooforum.org. Содержащий много полезных замечаний и примеров кода. Есть еще OpenOffice mail лист с ответами для разработчиков, но там мне так и не ответили).

Думаю смысла описывать все стороны разработки нет, поэтому остановлюсь на части непосредственно связанной с работой с OpenOffice. На этом этапе необходимо было сделать следующее: перебрать последовательно все объекты презентации ,инициировать на их значениях самодельный объект сущность презентации и экспортировать графические объекты (фон, фигуры, надписи, рисунки) в файлы изображений. Рассмотрим эти этапы более подробно.

Во первых чтобы вызывать сервисы OpenOffice через API нужно запустить его со специальными параметрами, хост на котором он запущен и порт который нужно слушать.

```
soffice -accept=socket,host=localhost,port=2002;urp;StarOffice.ServiceManager
```

Потом эти параметры используются в программе, для соединения с офисом. Т.е в принципе офис и программа работающая с ним могут находится на разных машинах, а это несомненно большой плюс.

К каждому элементу в OpenOffice прикреплено несколько объектов, к презентации слайды, к сладам фигуры. Последовательный перебор осуществляется с помощью интерфейса XIndexAccess

методами Object getByIndex(int) возвращает объект на ш месте

и int getCount() возвращает количество объектов

Например можно осуществить так :

```
// получаем слайд
```

```
XDrawPage page;
```

```
//Создаем объект для доступа по индексу
```

```
XIndexAccess ind = (XIndexAccess) UnoRuntime.queryInterface(
```

```

        XIndexAccess.class, page);
//Получаем конкретную фигуру
XShape newShape = (com.sun.star.drawing.XShape) UnoRuntime
        .queryInterface(com.sun.star.drawing.XShape.class, ind.getByIndex(i));

```

Любой графический объект в OpenOffice можно экспортировать в файл изображения, официально поддерживаются следующие форматы image /x-MS-bmp application/dxf application/postscript image /gif image /j peg image /png image /x-pict image /x-psx image /x-portable-bitmap image /x-portable-graymap image /x-portable-pixmap image /x-cmu-raster image /t arga image /tiff image /x-xbitmap image /x-xpixmap image /svg+xml хотя svg например работает только для слайда целиком, фигуры в него экспортировать нельзя.

Работает экспортер следующим образом: сначала получаем GraphicExportFilter через глобальный OpenOffice ServiceManager , потом используем XExporter интерфейс чтобы сказать фильтру какую страницу, фигуру или коллекцию фигур нужно экспортировать.

Экспортировать визуальный объект презентации в файл изображения можно следующим методом

```

public String converToImage(XInterface shape, String fileName, ImgType imgType) {
    try {

// Получаем объект GraphicExportFilter легковесный типа Proxy
        Object graphicExportFilter = xLocalServiceManager
                .createInstanceWithContext(
                    "com.sun.star.drawing.GraphicExportFilter",
                    xRemoteContext);
//Получаем реальный объект GraphicExportFilter
        XExporter xExporter = (XExporter) UnoRuntime.queryInterface(
            XExporter.class, graphicExportFilter);
//Получаем объект который будет обрабатывать фильтр
        XComponent xComp = (XComponent) UnoRuntime.queryInterface(
            XComponent.class, shape);
        xExporter.setSourceDocument(xComp);
//Создаем объект Файл в который будем записывать результат экспорта
        java.io.File destFile = new java.io.File(fileName + "." + imgType);
        StringBuffer destUrl = new StringBuffer("file:///");
        destUrl.append(destFile.getCanonicalPath().replace("\\", "/"));
// настройки для создания рисунка
        PropertyValue[] filterDatas = new PropertyValue[3];
        filterDatas[0]= new PropertyValue();
        filterDatas[0].Name="PixelWidth";
        filterDatas[0].Value=100;
        filterDatas[1]= new PropertyValue();
        filterDatas[1].Name="PixelHeight";
        filterDatas[1].Value=100;
        filterDatas[2]= new PropertyValue();
        filterDatas[2].Name="Translucent";
        filterDatas[2].Value=0;
        PropertyValue aProps[] = new PropertyValue[3];

```

```

        aProps[0] = new PropertyValue();
        aProps[1] = new PropertyValue();
        aProps[0].Name = "FilterName";
        aProps[0].Value = "EMF";
        aProps[1].Name = "URL";
        aProps[1].Value = destUrl.toString();
        aProps[2] = new PropertyValue();
        aProps[2].Name = "FilterOptions";
        aProps[2].Value = filterDatas;
//Создаем сам фильтр
        XFilter xFilter = (XFilter) UnoRuntime.queryInterface(
                XFilter.class, xExporter);
// Работает экспорт объектов в файл изображения
        xFilter.filter(aProps) .debug(xFilter.filter(aProps) ? "convert image"
                : "non convert image");
        return destUrl.toString();
    } catch (com.sun.star.uno.Exception e) {
        e.printStackTrace();
    } catch (java.lang.Exception e) {
        e.printStackTrace();
    }
}
return null;
}

```

Чтобы получить свойства объекта реализующего интерфейс XComponent (все графические компоненты его наследуют) нужно получить из него объект типа XPropertySet и использовать метод

Object `getPropertyValue(String propertyName)`

```

// Создаем легковесный Proxy объект слайд
XDrawPage xDrawPage = (com.sun.star.drawing.XDrawPage) UnoRuntime
        .queryInterface(com.sun.star.drawing.XDrawPage.class, ind
                .getByIndex(0));
// Создаем объект XpropertySet содержащий все атрибуты XDrawPage
        XPropertySet xPropertySet = (XPropertySet) UnoRuntime.queryInterface(
                XPropertySet.class, xDrawPage);
// Получаем значения конкретного свойства
Object o = getXproperties("Background",
        xPropertySet);
public static Object getXproperties(String name, XPropertySet xPropertySet){
    try {
        return xPropertySet.getPropertyValue(name);
    } catch (UnknownPropertyException e) {
        /.error("нет такой настройки "+name);
        return null;
    }
    catch (WrappedTargetException e){
        /.error("WrappedTargetException "+name);
        return null;
    }
}
}

```

В принципе разобравшись с этими примерами можно смело приступать к реализации собственного экспортера в любой другой формат. Хотя конечно же есть еще много подводных камней которые выясняются только при более детальном изучении структуры объектов OpenOffice. Например в java API для OpenOffice нет описания свойств для аргумента метода `XFilter.filter(PropertyValue pv)` и где их брать в официальной документации не написано, благо есть форум. Еще можно отметить продуманность структуры презентации она имеет общий стиль по умолчанию который применяется к каждому слайду и кроме того можно создавать стиль для каждого отдельного слайда, последний будет доминировать над общим.

На сегодня программа реализована примерно на 90%, отсталость сделать прозрачный фон у рисунков, в которые экспортируются фигуры, средствами OpenOffice похоже этого достичь не удастся, придется использовать сторонний конвертер изображений вроде JMagick. Еще конечно хочется экспортировать надписи не как изображения, а простым текстом, чтобы снизить трафик и увеличить быстродействие. Эта задача по идее тоже вполне решаемая, но на все нужно время, а здесь нужно разобраться со всеми стилями, списками, таблицами. Вобщем если работать в свободное время уйдет еще не один сезон.

Если у кого-то возникло желание расширить функциональность описанного конвертера или просто поэкспериментировать обращайтесь.