6.
アドオンパッケージ作成講座

JFEシステムズ
小野 誠 Ono Makoto
m-ono@jfe-systems.com

Copyright(C) 2006 Makoto Ono

この文書は、以下の利用条件の元で公開されています。

SOME RIGHTS RESERVED

クリエイティブコモンズ 帰属 2.1 日本

http://creativecommons.org/licenses/by/2.1/jp/

あなたは以下の条件に従う場合に限り、自由に

- 本作品を複製、頒布、展示、実演することができます。
- 二次的著作物を作成することができます。
- 本作品を営利目的で利用することができます。

あなたの従うべき条件は以下の通りです。

BY:

帰属. あなたは原著作者のクレジットを表示しなければなりません

- 再利用や頒布にあたっては、この作品の使用許諾条件を他の人々に明らかにしなければなりません。
- 著作[権]者から許可を得ると、これらの条件は適用されません。

上記によってあなたのフェアユースその他の権利が影響を受けることはまったくありません。

この文書は、Software Design 2006.5(技術評論社)に掲載した第 2 特集「OpenOffice.org で実現!次世代オフィススイート活用法」の原稿に加筆・修正したものです。原稿の公開にご協力いただいた Software Design 編集部に感謝します。

はじめに

4 章では OpenOffice.org の技術基盤である UNO の解説を行い，5 章では実際に存在するアドオンパッケージの使用例を紹介しました．6 章では，さらに進んでアドオンプログラム開発の一手法を説明します．ぜひ皆さんもお試しください．

具体的には，以下のような方針で記述していきます．

(1)開発言語は Java で

(2)独自知識はできるだけ少なく

(3)すぐに開発を始められるように

(1)について，Java を採用するのは私が Fio を開発するのに Java を用いたためですが，これによりアドオンは Windows でも Linux でも，OpenOffice.org の動作する環境であればどこでも動作するようになります．

(2)の「独自知識はできるだけ少なく」とは，OpenOffice.org アドオン開発のために覚えなければならないことを極力少なくするということです．UNO に代表される独自知識を縦横無尽に駆使するというのも一つの有効な戦略ですし，私がいくら「少なく」と言ったところで結局は OpenOffice.org の機能を使うためにある程度の学習は必要なのですが，OpenOffice.org のライブラリや構造に依存し過ぎないことを意識しておけば，単に学習量を減らせるだけでなく OpenOffice.org の仕様変更などの際にもアドオンへの影響を小さくできます．それに加えて「既存の Java 知識や既存の Java 資産を有効活用できる」というのが今回の作戦のポイントとなります．

最後に(3)ですが，本章は OpenOffice.org アドオン開発についての基礎知識や参考情報を提供することが目的なのではなく，読者がこれを読み終わった後で，すぐにでもアドオン開発をスタートできるようにするのが目標だということです．

そもそもアドオンとは？

OpenOffice.org を利用したプログラムには，大きく以下の 3 種類があります．

(1)マクロ

OpenOffice.org Basic などのスクリプト言語で書かれたマクロを実行することで機能を実現するものです．Excel VBA のようなもので，みなさんにも馴染み深いでしょう．

(2)OpenOffice.org サーバとの通信

OpenOffice.org をデーモンとして起動し，通信によってユーザ・アプリケーションから OpenOffice.org とやりとりするものです．必ずしもリモートマシン上の OpenOffice.org に接続しなければならないわけではありませんが，ソケット通信で OpenOffice.org を制御することが特徴となります．たとえば，一見 OpenOffice.org とは関係無さそうに見える普通のアプリケーションから OpenOffice.org のファイル出力機能を呼び出して，Excel や PDF や OpenDocument 形式のファイルを出力するといった使い方ができます．

(3)アドオン

Java，C++，OpenOffice.org Basic などで開発したプログラムを OpenOffice.org に登録することで，独自の機能を OpenOffice.org に追加するものです．今まで，OpenOffice.org として独自性の高い(2)についての情報はいろいろな機会に発信されてきましたが，よりシンプルで取り掛かり易いアドオンについての情報が不足していたように思います．

本章ではこれを扱います．

決め事と検証環境

作成していくプロジェクトの名前は"tansan"（**単**純な**サン**プルの意）とし，OpenOffice.org Calc へのアドオンとします．Writer などにアドオンを追加するときも，今回と全く同様の手法が使えます．

私が検証を行った環境は以下の通りです．

- OS：WindowsXP Home Edition
- OpenOffice.org：2.0.1
- Java：SUN JVM 1.5.0_06
- IDE：Eclipse 3.1.2

内容については Linux(Vine Linux 3.2)でも同様に行えることを検証していますが，本文の表記は上の環境に沿ったものとなりますので，他の環境を使用される方は適宜読みかえてください．

また，今回開発するアドオンの完成版（配布用パッケージ）は全てのソースファイルを含んだ形で SD 誌の Web サイト[1]および Fio の Web サイト[2]で公開しておりますので，参考のため，または開発用の雛型としてご活用ください．

開発の準備

開発環境のセットアップ

さっそくコーディングに入りたいところですが，開発のためにはいくつか特別な準備が必要です．これから順を追って説明していきますが，前提として OpenOffice.org および Java のインストールは完了しており，OpenOffice.org で Java を使用する設定もなされているものとしますので，まだ済んでいないという方はそちらを先に完了しておいてください（設定方法は 2 章のコラム参照）．

OpenOffice.org SDK のインストール

アドオンを開発するためには，専用の SDK（Software Development Kit）が必要です[3]．

OpenOffice.org 本家サイトからバージョン 2.0 版（680 版とも呼ばれます）の SDK をダウンロードしてください．

- 2.0 版
  http://download.openoffice.org/680/sdk.html
- SDK トップページ
  http://www.openoffice.org/dev_docs/source/sdk/

ダウンロードした zip ファイルを解凍して，適当な場所に置きます（そこを%SDK%と表記することにします．また，OpenOffice.org のインストール先を%OFFICE%と表記することにします）．

環境変数 PATH に以下の 2 ディレクトリを追加してください．

%SDK%¥windows¥bin

%OFFICE%¥program

注1: http://www.gihyo.co.jp/magazines/SD/support/200605
注2: http://seiza.dip.jp/machine/fio/developer/files/tansan.zip
注3: 実は，今回紹介した開発手法であれば SDK のインストールは必須ではありません．idl ファイルをコンパイルしたり，OpenOffice.org レジストリ情報を結合したりする必要がないからです．それでもあえてインストールした理由は，SDK に含まれるドキュメントは必須であることと，SDK に触れることが OpenOffice.org 理解を助けるためです．

これで開発に必要なプログラムが使用可能となります[4].

ライブラリ群に CLASSPATH を通す

OpenOffice.org にアクセスするためには,OpenOffice.org の API を利用します.

%OFFICE%¥program¥classes 以下の全ての jar ファイルに CLASSPATH を通してください.

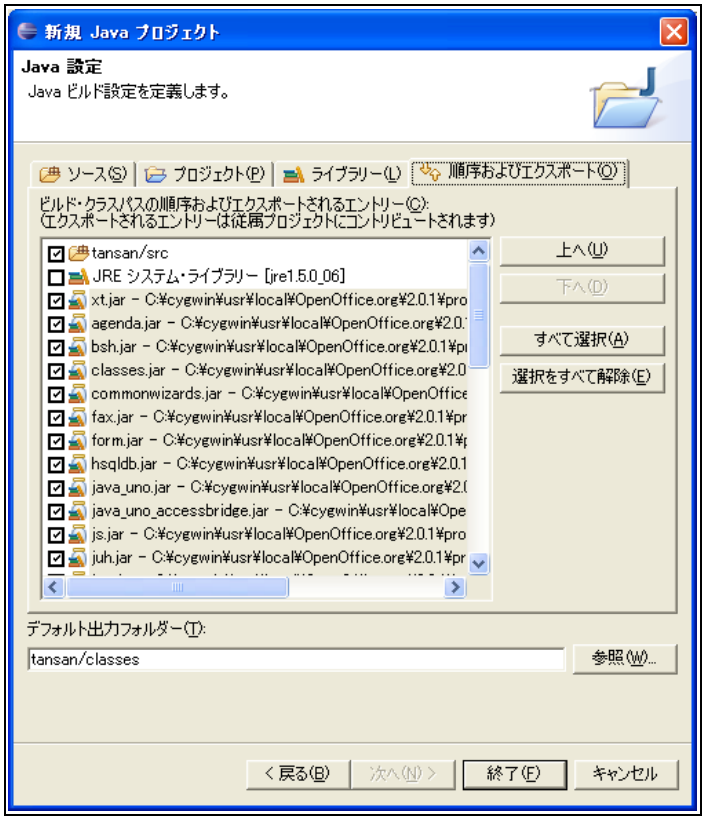

図 1:Eclipse での設定例

ドキュメントを準備する

次に,開発時に参照する各種のドキュメントを準備します.以下に示すドキュメント類にブックマークを張るなどして,必要な時にいつでも参照できるようにしておいてください.

開発者用ドキュメント

このドキュメントには UNO の概念や開発時の注意点,OpenOffice.org コンポーネント全体への詳細な記述が含まれており,OpenOffice.org の全体像を把握するためには欠かせません.

%SDK%¥docs¥DevelopersGuide¥以下に置かれている HTML ファイル群です(PDF 版も同梱されています).

ただし,残念なことに OpenOffice.org 2.0 SDK の文書は今のところ英語版しかありません[5].

注4: SDK にはインストールスクリプト(Windows ならバッチファイル,Linux ならシェルスクリプト)が同梱されていますので,これらを使ってインストール(環境変数の設定)を行っても良いでしょう.ただしその場合は,ここで述べた手順以上の準備が必要になる場合があります.

注5: 1.1 版 SDK には有志による日本語訳(http://www.opendt.org/pukiwiki/pukiwiki.php?Translation)が存在するのですが,かなり機械翻訳に近いもので読み辛さを感じます.

私として最もお勧めするのは，Sun Microsystems 社が OpenOffice.org をベースに開発している StarSuite の日本語文書です（http://www.sun.com/software/star/starsuite/sdk/index.jsp）．

サイト自体は英語ですが，日本語版の SDK をダウンロードすることで日本語訳を入手可能です．さすがと言うべきか，企業が提供する翻訳ですので質は非常に高いです．この文書の "StarSuite" を"OpenOffice.org"と読み替えれば，ほぼそのままで通用します．またこのことは，OpenOffice.org のアドオンが StarSuite のアドオンとしても機能するということを示しています[6]．

このドキュメントは画像や CSS を多様しているため，ブラウザで表示するために非常に多くのメモリと CPU を消費することには注意してください．

IDL リファレンス

OpenOffice.org にアドオンプログラムからアクセスするための API のドキュメントです．IDL という抽象化された概念で定義されているため，C++開発者も Java 開発者も同じこのドキュメントを読むことになります．

プログラミング中に「セルからデータを取得するにはどうすれば良いか？」「自動幅揃えを実行するにはどうすれば良いか？」といった疑問が起こった場合，最終的にはこのドキュメントを読むことになります．

残念なことに日本語訳が存在しないようなのですが，普段 Java の API 文書を読みなれている人であれば高い英語力は必要ありません．

%SDK%¥docs¥common¥ref¥com¥sun¥star¥module-ix.html です．

その他の有益な情報たち

上記の基本的な文書以外にも，ためになる文書は多数公開されています．特に有益と思われる情報をリストしておきますので，活用してください．

- StarSuite SDK によるアドオン開発
  http://sdc.sun.co.jp/solaris/starsuite/
- OOoForum.org（英語）
  http://www.oooforum.org/
- OpenOffice.org 開発 Wiki
  http://hermione.s41.xrea.com/pukiwiki/pukiwiki.php?FrontPage
- StarSuite 7 ドキュメント
  http://jp.sun.com/products/software/starsuite/7/docs/

また，前記の SDK に含まれるドキュメント類は英語版であればオンラインでも読むことができます（http://api.openoffice.org/）．

プログラミング開始！

プログラミングの流れ

さあ準備が整いました．プログラミングを始めましょう．

独自メニューをクリックしてから処理が実行されるまでの流れと，これから作成していくファイル群の関連をまとめました．とりあえず大まかな流れを掴んでいただければ充分です．詳細についてはこれからプログラミングしながら，この図 2 に立ち戻って解説します．

注6: StarSuite SDK を導入しようとすると SDK のインストーラを走らせることになります（ドキュメントだけを取り出すことはできません）が，特に問題はありません．

また,ファイルを 3 種類に分けて図示していますが,以下のような分類となっています.

(1)主要な開発対象

(2)開発時に簡単な追記/編集が必要となるファイル

(3)1 度書いたらめったに更新する必要の無いファイル

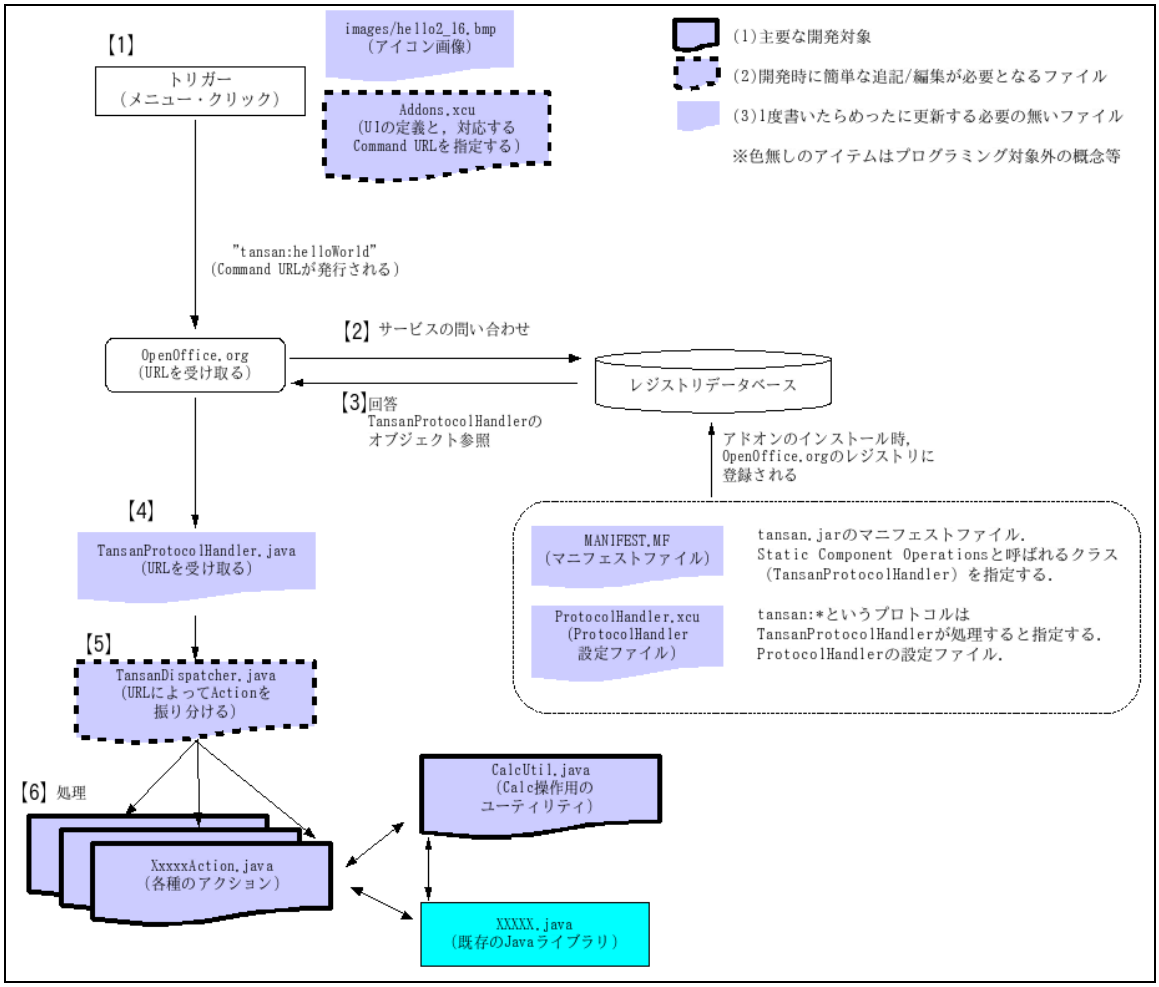

図 2:プログラミングの流れ

こうして見ると,始めはたくさんの見慣れないファイルを書かなければいけないように見えるものの,一旦一通りのファイルが揃った後ではアプリケーションロジックを実装する Java ファイルに注力していけるようになるのが分かると思います.実際のところ,今回の開発方法をとる限り IDL は記述の必要さえなく,ProtocolHandler の設定ファイルなどは型通りのもので良いので,特に新しい文法などを覚える必要はありません[7].

ディレクトリ構成

プロジェクトのディレクトリ構成

tansan プロジェクトのディレクトリ構成は以下の通りとします.以下に当てはまらないファイルはプロジェク

注7: では IDL を書くことの意義はどこにあるかですが,IDL ファイルからはレジストリ情報を生成することができ,それを図 2 のレジストリデータベースに登録することで「OpenOffice.org の API がそうであるように」言語に依存しない形で C++や Basic や Java から,開発した機能を利用することが可能になります.

ト直下に置くことにします.

classes(class ファイル)

images(アイコンファイル)

javadoc(javadoc 出力先)

release(配布用パッケージの出力先)

src(java ファイル)

メニュー項目を付ける

アドオンを起動するために,Calc に独自メニューを追加します.図 2 では「トリガー」と記述した部分です.UI を追加するには Addons.xcu というXML ファイルを記述します.

テキスト 1:Addons.xcu

```
<?xml version='1.0' encoding='UTF-8'?>
<oor:component-data xmlns:oor="http://openoffice.org/2001/registry" xmlns:xs="http://www.w3.org/2001/XMLSchema" oor:name="Addons" oor:package="org.openoffice.Office">
 <node oor:name="AddonUI">
  <!-- ********** add menu to the top level menu bar ********************************** -->
  <node oor:name="OfficeMenuBar">
   <node oor:name="tansan" oor:op="replace"> ★1
    <prop oor:name="Title" oor:type="xs:string">
     <value>Tansan(~3)</value> ★2
    </prop>
    <prop oor:name="Context" oor:type="xs:string">
     <value>com.sun.star.sheet.SpreadsheetDocument</value> ★3
    </prop>
    <prop oor:name="Target" oor:type="xs:string">
     <value>_self</value>
    </prop>

    <!-- ********** add submenu to the menu ************************************* -->
    <node oor:name="Submenu"> ★4
     <!-- ************************************************************************ -->
     <node oor:name="m0100" oor:op="replace"> ★5
      <prop oor:name="URL" oor:type="xs:string">
       <value>tansan:helloWorld</value> ★6
      </prop>
      <prop oor:name="Title" oor:type="xs:string">
       <value>Hello World(~H)</value>
```

```
 </prop>
 <prop oor:name="Target" oor:type="xs:string">
  <value>_self</value>
 </prop>
</node>

<!-- ========================================================================== -->
<node oor:name="m0110" oor:op="replace">
 <prop oor:name="URL" oor:type="xs:string">
  <value>private:separator</value> ★7
 </prop>
</node>

<!-- ************************************************************************** -->
<node oor:name="m0210" oor:op="replace">
 <prop oor:name="Title" oor:type="xs:string">
  <value>サブメニュー(~S)</value> ★8
 </prop>
 <prop oor:name="Target" oor:type="xs:string">
  <value>_self</value>
 </prop>

 <!-- ********************************************************************** -->
 <node oor:name="Submenu"> ★9
  <!-- ****************************************************************** -->
  <node oor:name="m0220" oor:op="replace">
   <prop oor:name="URL" oor:type="xs:string">
    <value>tansan:sub_HelloWorld2</value>
   </prop>
   <prop oor:name="Title" oor:type="xs:string">
    <value>Hello World2(~H)</value>
   </prop>
   <prop oor:name="ImageIdentifier" oor:type="xs:string"> ★10
    <value>%origin%/images/hello2</value>
   </prop>
   <prop oor:name="Target" oor:type="xs:string">
    <value>_self</value>
   </prop>
```

```
        </node>
       </node>
      </node>
     </node>
    </node>
   </node>
  </node>
</oor:component-data>
```

★1:で,メニューバーのトップレベルに名前を付けています[oor:name="tansan"].

ここは任意で良いのですが,よそのファイルとバッティングしないように,Java のパッケージ名を使用しておきましょう.

★2:メニューバートップレベルの Title に付けられた[Tansan(~3)]が,Calc のメニューバーに追加表示される文字列となります.

[~3]というのはニーモニックの指定で,Alt+3 というキーボードショートカットでこのメニューにアクセス可能にしています.

★3:[com.sun.star.sheet.SpreadsheetDocument]と,このメニューが追加されるのが Calc であることが指定されています.

Writer などにメニューを追加したい場合は下記を参照してください.

表 1:Writer などにメニューを挿入

Writer	com.sun.star.text.TextDocument
Impress	com.sun.star.presentation.PresentationDocument
Draw	com.sun.star.drawing.DrawingDocument

★4:[Submenu]と指定することで,Tansan メニュー内のアイテムが追加できます.

★5:ここからのひとまとまりが一つのメニュー項目を定義しているわけですが,[oor:name="m0100"]という名前に注意してください.これも実は任意の文字列で構わないのですが,以下の 2 点に注意する必要があります.

(1)スペースは許されない

ダブルクォートしているのに関わらず,ここではスペースは許されません.スペースがあるとメニューが表示されなくなったりするなど,トラブルのもとです.

(2)メニュー項目のソート順を表している

実際にメニューが表示されるときの表示順は,ここの文字列のソート順で制御されます.ファイル内での記述順は関係ありません.

上記のような理由から,私はメニューを表す"m"と数字を付けた名前に統一しています.数字はある程度飛び番にすることで,後でメニューを追加したくなったときに対応しています.

★6:[Hello World(~H)]という文字列のメニューに対して,[tansan:helloWorld]という Command URL が対応していることを記述しています.

Command URL の表記法ですが,<パッケージ名>:<コマンド名>というものをお勧めします.パッケージ名

が org.openoffice.ja.pkg というものであれば，org.openoffice.ja.pkg:commandName といった要領です．この Command URL は後で作成する ProtocolHandler 類のファイルで使用します．

★7：これは特殊な URL で，[private:separator]と入力しておくとメニューバー内に区切り線を入れることができます．

★8：メニュー文字列にはもちろん日本語も使用できます．1 行目で encoding を UTF-8 と指定していますので，日本語を使用する場合はこのファイルを UTF-8 で保存することを忘れないでください（ファイルを半角英数字だけで記述する場合は特に意識しなくても大丈夫です．半角英数字は，Shift_JIS も EUC-JP も UTF-8 も同一のコードになるためです）．

また，この node には URL の指定がないことに注意してください．この項目は，以降のサブメニュー項目を入れるための箱に過ぎないためです．

★9：上述の通り，このように Submenu をネストすることで自由にサブメニュー（階層のあるメニュー）を作成できます．

★10：[oor:name="ImageIdentifier"]この記述は，メニューにイメージアイコンを付加するためのものです．下の[%origin%/images/hello2]という value 指定にも注目してください．ここについては続いて説明します．

イメージアイコンの付加

上で[%origin%/images/hello2]と指定しましたが，実際にアイコン画像ファイルを作成しなければいけません．

OpenOffice.org は自動的に"_16.bmp"という文字列を補完してアイコンファイルを探しますので，images ディレクトリには hello2_16.bmp という 1 辺 16 ピクセルのビットマップファイルを置く必要があります．

GIMP などのツールでアイコン画像を作成し，images ディレクトリに置いてください[8]．

最初のパッケージ

Addons.xcu とアイコン画像だけでもアドオンパッケージとして取り込みが可能です．

Addons.xcu と images ディレクトリを圧縮して tansan.zip ファイルを作成してみましょう．

ここでは，JDK に付属している jar コマンドで zip ファイルを作成します．コマンドプロンプトで tansan プロジェクトディレクトリに移動し，以下のコマンドを実行します．

```
> jar cf release\tansan.zip images Addons.xcu
```

これで release ディレクトリに tansan.zip ファイルができます．OpenOffice.org のパッケージマネージャで取り込み，OpenOffice.org を再起動すると，図 3 のようになります．

メニューに対応した実装がないため項目はグレイ表示されていますが，意図した通りに Tansan メニューが追加されました．

注8: ツールバーにアイコンを付ける場合，大きいアイコンを使用するユーザのために 1 辺 26 ピクセルの hello2_26.bmp も必要となります．また，ツールバーにアイコンを付ける場合の書き方は（テキスト A）のようになります．"OfficeMenuBar"と同じレベルの箇所に挿入してください．

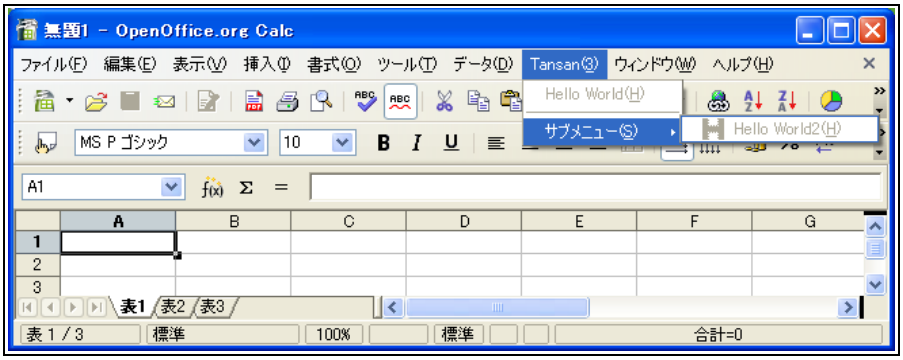

図 3:追加されたメニュー

テキスト A:ツールバーにアイコンを付ける

```
<node oor:name="OfficeToolBar">
 <node oor:name="tansan" oor:op="replace">
 <node oor:name="m2000" oor:op="replace">
  <prop oor:name="Title" oor:type="xs:string">
   <value>Hello!</value>
  </prop>
  <prop oor:name="ImageIdentifier" oor:type="xs:string">
   <value>%origin%/images/hello2</value>
  </prop>
  <prop oor:name="URL" oor:type="xs:string">
   <value>tansan:helloWorld</value>
  </prop>
  <prop oor:name="Target" oor:type="xs:string">
   <value>_self</value>
  </prop>
  <prop oor:name="Context" oor:type="xs:string">
   <value>com.sun.star.sheet.SpreadsheetDocument</value>
  </prop>
 </node>
 </node>
</node>
```

OpenOffice.org に URL とプログラムの対応を教える

図 2 を見てください.ここまでの作業で Calc のメニューバーに tansan を追加することができましたが,Hello World メニューをクリックしたときに発行される"tansan:helloWorld"という Command URL を受け取った後でどのプログラムを実行するのかを,OpenOffice.org に教えてやらなければいけません.

ProtocolHandler.xcu

まずは ProtocolHandler の設定ファイルを書きます.このファイルでは Command URL に対応する ProtocolHandlerを指定します.

テキスト 2:ProtocolHandler.xcu

```
<?xml version='1.0' encoding='UTF-8'?>
<oor:component-data
 oor:name="ProtocolHandler"
 oor:package="org.openoffice.Office"
 xmlns:oor="http://openoffice.org/2001/registry"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

 <node oor:name="HandlerSet">
  <node oor:name="tansan.uno.TansanProtocolHandler" oor:op="replace"> ★1
   <prop oor:name="Protocols" oor:type="oor:string-list">
    <value>tansan:*</value> ★2
   </prop>
  </node>
 </node>
</oor:component-data>
```

★1:このアドオンの ProtocolHandler を指定します.以降で作成する ProtocolHandler クラスをパッケージ名から記述します.

★2:上で指定した ProtocolHandler が処理する Command URL を指定します."*"によるワイルドカードが有効ですので,tansan の全てのアクションを処理するように指定しています.

MANIFEST.MF

以降で作成する Java のプログラムは,jar ファイルに固めてから配布用パッケージに含めることとなります.その jar ファイルのマニフェストファイルがこれです.

テキスト 3:MANIFEST.MF

```
Manifest-Version: 1.0
RegistrationClassName: tansan.uno.TansanProtocolHandler ★1
```

★1:RegistrationClassName として,フルパッケージ名で TansanProtocolHandler クラスを指定しています.

ここで,RegistrationClassName の指定は,厳密には「ProtocolHandler クラスを」指定しているわけではなく,「Static Component Operations と呼ばれるクラスを」指定しています(TansanProtocolHandler には ProtocolHandler と Static Component Operations の両方の機能を持たせています).

Static Component Operations はアドインパッケージのインストール時に実行され,OpenOffice.org のレジストリ(4章参照)にアドオンの情報を登録します.これにより,OpenOffice.org は ProtocolHandler.xcu に指定された TansanProtocolHandler が Java のクラスであり,このアドオンパッケージに含まれているということを知ることができます.

ProtocolHandler の作成

ついに Java の世界に制御が渡されました.ProtocolHandler は OpenOffice.org とアドオンプログラムとの接続を仲介する重要なクラスです.ComponentBase クラスを継承し,XServiceInfo,XDispatchProvider,XInitialization の 3 つのインターフェイスを実装します.これらのインターフェイスを実装しなければならないことからソースは長くなってしまいますが,それぞれのメソッドはおまじないのようなものですので,まずはここにある通り記述してください.オリジナルのアドオンを開発する時には,ソース中"Tansan"や"tansan"と書かれた部分について書き換えて使ってください.

テキスト 4:ProtocolHandler

```
package tansan.uno;

import com.sun.star.lib.uno.helper.*;
import com.sun.star.lang.*;
import com.sun.star.uno.*;
import com.sun.star.frame.*;
import com.sun.star.registry.*;
import com.sun.star.comp.loader.*;

/**
 * Addons.xcu に記述されるメニュー項目と紐付いたアクションの URL(tansan:)を
 * 監視し、TansanDispatcher に発送するハンドラクラス。
 */
public class TansanProtocolHandler extends ComponentBase implements XServiceInfo, XDispatchProvider, XInitialization{

    private XComponentContext context;

    /**
     * コンストラクタ:TansanManager の初期化、context の保存等。
     * @param context
     */
    public TansanProtocolHandler(XComponentContext context){ ★1
        this.context = context;
    }

    protected void finalize(){
    }
```

```
/* XInitialization implementation ************************************************************/
private XFrame frame;
public void initialize(Object[] obj) throws com.sun.star.uno.Exception{
    if(obj.length > 0){
        frame =(XFrame)UnoRuntime.queryInterface(XFrame.class, obj[0]);
    }
}

/* XDispatchProvider implementation **********************************************************/
public XDispatch[] queryDispatches(DispatchDescriptor[] descs){
    XDispatch[] dispatcher = new XDispatch[descs.length];

    for(int i = 0; i < descs.length; i++){
        dispatcher[i] = queryDispatch(descs[i].FeatureURL, descs[i].FrameName, descs[i].SearchFlags);
    }
    return dispatcher;
}

public XDispatch queryDispatch(com.sun.star.util.URL url, String targetFrameName, int searchFlags){ ★2
    if(url.Protocol.equals("tansan:")){
        return new TansanDispatcher(context, frame);
    }else{
        return null;
    }
}

/* XServiceInfo implementation ***************************************************************/
static final String SERVICENAME = "com.sun.star.frame.ProtocolHandler";

public String[] getSupportedServiceNames(){
    return new String[] {SERVICENAME};
}

public String getImplementationName(){
    return getClass().getName();
}
```

```
    public boolean supportsService(String serviceName){
        if(serviceName.equals(SERVICENAME)){
            return true;
        }else{
            return false;
        }
    }

    /* static component operations **************************************************************/
    /* MANIFEST.MF で指定したクラスでは、これらのインターフェイスを実装しなければならない */
        public static XSingleServiceFactory __getServiceFactory(String implName, XMultiServiceFactory multiFactory,
XRegistryKey regKey){ ★3
        XSingleServiceFactory xSingleServiceFactory = null;

        if(implName.equals(TansanProtocolHandler.class.getName())){
                                    xSingleServiceFactory = FactoryHelper.getServiceFactory(TansanProtocolHandler.class,
TansanProtocolHandler.SERVICENAME, multiFactory, regKey);
        }

        return xSingleServiceFactory;
    }

    public static boolean __writeRegistryServiceInfo(XRegistryKey regKey){ ★3
                                return FactoryHelper.writeRegistryServiceInfo(TansanProtocolHandler.class.getName(),
TansanProtocolHandler.SERVICENAME, regKey);
    }
}
```

これを書き終えた段階では次に作成する TansanDispatcher クラスがまだ無いためにコンパイルエラーとなってしまいますが,それは構いません.

アドオンの動作を理解するために特に重要な 3 点を説明します.

★1:まずはコンストラクタです.XComponentContext のインスタンスを受け取って,それを private フィールドに保持しています.追加メニューなどからアドオンが起動されると,まずは MANIFEST.MF で指定されたこの ProtocolHandler クラスのオブジェクトが生成されます.もしアドオン全体の初期化処理(設定ファイルの読み込みなど)が必要になった場合は,このクラスのコンストラクタから処理を呼んでやるのが良いでしょう.

★2:次に queryDispatch メソッドです.これは XDispatchProvider インターフェイスで定義された抽象メソッドの実装ですが,メニューがクリックされた時に呼ばれるのはこのメソッドです.context と frame(それぞれメニュー起動時の OpenOffice.org/Calc 画面を表すオブジェクトです)が引数で渡された上で TansanDispatcher オブジェクトが生成されていることを確認してください.これについては次に説明します.

★3:MANIFEST.MF の説明で言及した Static Component Operations とは,具体的にはこれらのメソッドを指します.

Dispatcher の作成

OpenOffice.org は ProtocolHandler の queryDispatch メソッドを経由して Dispatcher と呼ばれるこのクラスのオブジェクトを生成し，dispatch メソッドを呼びます．OpenOffice.org が自動的に実行するのはここまでです．

テキスト 5:Dispatcher

```
package tansan.uno;

import tansan.action.*;

import com.sun.star.frame.*;
import com.sun.star.beans.*;
import com.sun.star.uno.*;
import com.sun.star.util.*;

/**
 * TansanProtocolHandler に観測された Tansan の URL メッセージに対して、
 * 適切な処理をするクラス/メソッドへと振り分ける。<br>
 */
public class TansanDispatcher implements XDispatch{

    private XComponentContext context;
    private XFrame frame;


    /**
     * コンストラクタ：context と frame が渡されてくるので private フィールドに保持
     */
    public TansanDispatcher(XComponentContext context, XFrame frame){
        this.context = context;
        this.frame = frame;
    }

    protected void finalize(){
    }

    public void addStatusListener(XStatusListener l, URL url){
    }
    public void removeStatusListener(XStatusListener l, URL url){
    }
```

```
    /**
     * URL が分かるので、ここで分岐させる
     */
    public void dispatch(URL url, PropertyValue[] args){
        try{
            if(url.Complete.equals("tansan:helloWorld")){
                HelloWorldAction action = new HelloWorldAction();
                action.doAction(context, frame);
            }else if(url.Complete.equals("tansan:sub_HelloWorld2")){
//              Sub_HelloWorld2Action action = new Sub_HelloWorld2Action();
//              action.doAction(context, frame);
            }
        }catch(java.lang.Exception e){
            ;
        }catch(java.lang.Error e){
            ;
        }
    }
}
```

引数 URL には"tansan:helloWorld"など，Addons.xcu に記述した Command URL が渡されてきますので，URL によって分岐してそれぞれの処理を行っていきましょう．ここで，コンストラクタで渡された XComponentContext と XFrame をアクションに渡しておくようにします．私たちが開発するアクションから Calc のセルなどにアクセスする時は，全てこの 2 つのオブジェクトが元になるというわけです．

また，例外（Exception だけでなく Error も）を catch しているところに注目してください．ここでは例外処理の記述を省略していますが，実際には最低でもログ出力などを実施することになると思います．もちろん本来であれば，例外は各アクションクラスの中で適切に処理されているはずですが，それでも捉えきれなかった例外を OpenOffice.org まで throw しない点が重要です．もしここで catch せず，OpenOffice.org の制御下まで例外が届いてしまうと，OpenOffice.org は異常終了します．

さらに，Exception や Error がフルパッケージ名で記述されているのも違和感があると思います．これは，OpenOffice.org のライブラリに同名のクラスが作成されているためです．正直言ってタイピングに手間がかかるだけなので，私は API 設計者の不手際だと思っています．

独自アクションの作成と，共通部品の作成

アクションの作成

ここからはアプリケーションロジックですので，それぞれ自由にプログラミングしていきましょう．図 2 の【6】にある通り，一度 Java に制御が渡されてしまえばこちらのものです．手元にある Java のライブラリを駆使してどんな処理でもすいすいと書いていけるでしょう．

今回のサンプルでは，実質 4 行で Calc のセルに「Hello! user-name」と出力しています．

★1：CalcUtil という共通部品を使って，新しい Calc ドキュメントを作成しています．

★2:最初のシートを取得しています.

★3:出力するメッセージです.ユーザ名の取得には Java の機能を利用しています.

★4:最後に,★2 で取得したシートに★3 で作成した文字列を出力しています.

テキスト 6:アクションの作成

```
package tansan.action;

import tansan.common.*;
import com.sun.star.uno.*;
import com.sun.star.frame.*;
import com.sun.star.sheet.*;

public class HelloWorldAction{
    public void doAction(XComponentContext context, XFrame frame){
        try{
            XSpreadsheets sheets = CalcUtil.getNewSpreadsheets(context); ★1
            XSpreadsheet sheet = CalcUtil.getSpreadsheetByIndex(sheets, 0); ★2
            String msg = "Hello! " + System.getProperty("user.name"); ★3
            CalcUtil.setCellString(sheet, 1, 1, msg); ★4
        }catch(java.lang.Exception e){
            ;
        }
    }
}
```

XSpreadsheet インターフェイスなどは OpenOffice.org の API です.いくら依存性の低いプログラムを意識しても,OpenOffice.org とデータをやりとりするからには最低限の OpenOffice.org API プログラミングが必要になります.

共通部品の作成

今回作成した 3 つのメソッドを見てください.典型的な OpenOffice.org プログラミングです.

シートを取得したりセルにデータをセットしたりといった一般的な処理は共通部品にまとめてしまいましょう.それによって共通化と再利用ができるだけでなく,アプリケーションロジックから OpenOffice.org API を遠ざけておけます.

今回の CalcUtil のように OpenOffice.org とやりとりする部品を書くときには,開発者用ドキュメントと IDL リファレンスを参照します.また,Fio の中にも同名の CalcUtil.java があり,こちらはすでに多くの機能が実装済みですので参考にしてください.

この中では,getNewSpreadsheets メソッドの 1 行目(★1)に注目してください.

ComponentContext から ServiceManager を取得し,ServiceManager の createInstance にサービス名を渡すことでオブジェクトを取得しています.4 章の「UNO の概念と用語」で解説されている通り,OpenOffice.org の API が特定のプログラミング言語に依存しないという UNO の真価が発揮された箇所です.

ここまでの作業でプログラミングは完了です.java ファイルがエラー無くコンパイルされることを確認してください.

テキスト 7:共通部品の作成

```
package tansan.common;

import com.sun.star.frame.*;
import com.sun.star.uno.*;
import com.sun.star.sheet.*;
import com.sun.star.text.*;
import com.sun.star.table.*;
import com.sun.star.beans.*;
import com.sun.star.lang.*;
import com.sun.star.container.*;

public class CalcUtil {
    /** XComponentContext から、新規 calc 文書の XSpreadsheets オブジェクトを返す。*/
    public static XSpreadsheets getNewSpreadsheets(XComponentContext context) throws com.sun.star.uno.Exception{
        Object desktop = context.getServiceManager().createInstanceWithContext("com.sun.star.frame.Desktop", context); ★1
        XComponentLoader loader =(XComponentLoader)UnoRuntime.queryInterface(XComponentLoader.class, desktop);
                XComponent component = loader.loadComponentFromURL("private:factory/scalc", "_blank", 0, new PropertyValue[0]);
            XSpreadsheetDocument doc =(XSpreadsheetDocument)UnoRuntime.queryInterface(XSpreadsheetDocument.class, component);
        return doc.getSheets();
    }

    /** XSpreadsheets から指定された index の XSpreadsheet を返す */
    public static XSpreadsheet getSpreadsheetByIndex(XSpreadsheets sheets, int index)
            throws com.sun.star.lang.IndexOutOfBoundsException, WrappedTargetException{
        XIndexAccess ia =(XIndexAccess)UnoRuntime.queryInterface(XIndexAccess.class, sheets);
        return(XSpreadsheet)UnoRuntime.queryInterface(XSpreadsheet.class, ia.getByIndex(index));
    }

    /** シートの(列、行)位置のテキストを設定する。*/
            public static void setCellString(XSpreadsheet sheet, int column, int row, String string) throws com.sun.star.lang.IndexOutOfBoundsException{
        XCell cell = sheet.getCellByPosition(column, row);
        XText cellText =(XText)UnoRuntime.queryInterface(XText.class, cell);
        cellText.setString(string);
    }
}
```

配布用パッケージの作成

プログラミングは完了しましたので,配布用にパッケージングします.具体的には以下のようにファイルを固めます.

tansan.zip

- Addons.xcu
- ProtocolHandler.xcu
- tansan.jar
- images¥hello2_16.bmp

まずはここに現れる tansan.jar を作成しなければいけません.

release ディレクトリに classes¥tansan と MANIFEST.MF をコピーし,そこで以下のコマンドを実行します.

```
> jar cvfm tansan.jar MANIFEST.MF tansan
```

次に,他のファイルとまとめます.同じディレクトリに Addons.xcu,ProtocolHandler.xcu,images ディレクトリをコピーし,以下のコマンドを実行します.

```
> jar cf tansan.zip tansan.jar Addons.xcu ProtocolHandler.xcu images
```

さあ,tansan.zip をパッケージマネージャで取り込んでテストしてみましょう.正常にメッセージは出力されましたか?

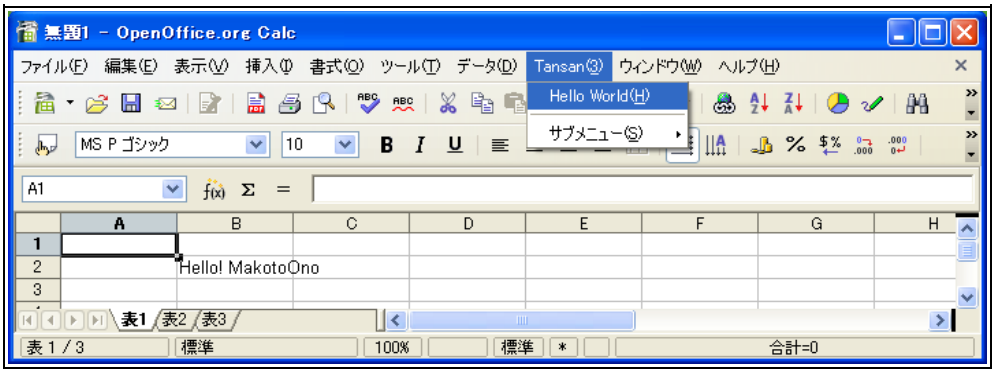

図 4:メッセージが表示される

機能の追加

図 2 を見てください.機能追加の手順は以下のようになります.

(1)Addons.xcu に定義を追加する

(2)必要ならアイコン画像を作成する

(3)Dispatcher の URL による分岐を追加する

(4)アプリケーションロジックの開発

tansan メニューの HelloWorld2 がまだ実装されていませんので,クリックしても何も起こりません.ここは読者にお任せしますので,何か処理を追加してみてください.機能追加時にはアプリケーションロジックだけに注力できることが実感できると思います.

おわりに

いかがでしょうか.OpenOffice.org 上にアドオンプログラムを載せ,Calc のデータをコントロールするというのが意外に簡単だと思っていただけましたか?

Fio では JDBC を使ってデータベースと接続し,Calc のスプレッドシートをデータベースのテーブルであるかのように利用しています.データベースアクセスツールとしても,Calc の高度な編集機能はとても役立ちます.

OpenOffice.org に機能を付加(add-on)するという発想だけではなく,自分が何かアプリケーションを開発しようと思った時に「OpenOffice.org の機能を利用したらもっと便利なものが容易に実現できないか」と発想してみるのも良いかもしれません.そんなふうに思っていただける人が現れることを,期待しています.

コラム:そして Fio でつながった

コンピュータ関連では 2 つの「物語」が私には印象的です.

ひとつはインターネット.HTTP が世界をリンクし始めた頃にブラウザの青く光る文字をクリックした人たちは「世界が変わる」という予感を共有したのではないでしょうか.

もうひとつは Linux.北欧の一学生が開発を始めた OS が,何の報酬があるわけでもないのに無数の協力者を得て世界中に普及した事実.

--

去年,私は仕事上で困難に直面していました.入り組んだ事情があり,そこでは今まで親しんでいたデータベースアクセスツールが使用できなかったのです.そんな時,OpenOffice.org と Java のプログラムを連携させられるということをどこかで知りました.ほとんど同時に「これは使えるかもしれない」という直感がありました.

プログラマーなら誰でも一度は「フリーウェアを作ってみようか」なんて思ったことがあると思います.でも,実際に作って公開している人は多くありません.きっと「きっかけ」の問題なのだろうと思います.

私には動機があり,少しのアイディアがあり,自分の技術と知識で実現できる範囲とそれだけでは届かない領域がぼんやりとは見えていました.OpenOffice.org と出会い,自分の求めることが自分で実現できそうだと思えるようになった時,私は疲れきった深夜にも少しずつなら開発を続けることができる自分を発見しました.そして,自分のプログラムが OpenOffice.org とつながり,データベースとつながり,検索結果が目の前のシートに出力された時,自分がしていることとインターネットや Linux の物語がほんの少しつながったような気がしました.