It.oduestudio.openoffice

JAVA API di connessione ad OpenOffice

Nicola Muratori, 2004

1 Abstract

In questo documento si descriverà la realizzazione di una API (Application Program Interface) in grado di mettere a disposizione al programmatore semplici e fruibili funzionalità di creazione di documenti di diversa tipologia (videoscrittura, fogli di calcolo, presentazioni) tramite l'applicazione OpenOffice.org.

L'argomento verrà trattato con le tecniche classiche dell'ingegneria del software, in particolare utilizzando la metodologia 'Object Oriented Analysis and Design' Voce bibliografia per quello che riguarda la definizione del problema e la definizione delle classi.

1.1 Cos'è Openoffice

La descrizione del progetto openoffice (www.openoffice.org) è reperibile all'indirizzo http://www.openoffice.org/about.html. A seguito si riportano i punti salienti:

> Project
>
> Mission Statement
>
> To create, as a community, the leading international office suite that will run on all major platforms and provide access to all functionality and data through open-component based APIs and an XML-based file format.
>
> Historical background
>
> *StarDivision, the original author of the StarOffice suite of software, was founded in Germany in the mid-1980s. It was acquired by Sun Microsystems during the summer of 1999 and StarOffice 5.2 was released in June of 2000. Future versions of StarOffice software, beginning with 6.0, have been built using the OpenOffice.org source, APIs, file formats, and reference implementation.*
>
> *The OpenOffice.org source code initially includes the technology which Sun Microsystems has been developing for the future versions of StarOffice(TM) software. The source is written in C++ and delivers language-neutral and scriptable functionality, including Java(TM) APIs. This source technology introduces the next-stage architecture, allowing use of the suite as separate applications or as embedded components in other applications. Numerous other features are also present including XML-based file formats and other resources.*
>
> *A FAQ addresses the changing differences between OpenOffice.org and StarOffice.*
>
> Product Description

OpenOffice.org is both an Open Source product and a project. The product is a multi-platform office productivity suite. It includes the key desktop applications, such as a word processor, spreadsheet, presentation manager, and drawing program, with a user interface and feature set similar to other office suites. Sophisticated and flexible, OpenOffice.org also works transparently with a variety of file formats, including those of Microsoft Office.

Available in 25 languages with more being constantly added by the community. OpenOffice.org runs stably and natively on Solaris, Linux (including PPC Linux), and Windows. Additional ports, such as for FreeBSD, IRIX, and Mac OS X, are in various stages of completion.

Written in C++ and with documented APIs licensed under the LGPL and SISSL Open Source licenses, OpenOffice.org allows any knowledgeable developer to benefit from the source. And, because the file format for OpenOffice.org is XML, interoperability is easy, making future development and adoption more certain.

1.2 Obiettivi

1. Migliorare la fruizione delle API di connessione ad OpenOffice, pensate principalmente per essere portabili su diverse piattaforme e su diversi linguaggi.

2. Offire semplici metodologie di manipolazione di documenti da applicazioni java, nascondendo allo sviluppatore la complessità del framework UNO.

1.3 Stato attuale delle API – esempio di una chiamata

Al momento attuale le API di connessione ad OpenOffice sono realizzate secondo una tecnica a 'servizio'. Senza addentrarsi nei particolari, descriveremo le principali funzionalità del framework UNO, alla base del package in realizzazione (il seguente testo è tratto dalla Developer Guide dell'OpenOffice SDK, reperibile sul sito www.openoffice.org):

Le premesse delle funzionalità e dell'ambito di utilizzo di UNO sono ambiziose; l'implementazione di conseguenza ne riflette in parte la apertura e la complessità.

The goal of UNO (Universal Network Objects) is to provide an environment for network objects across programming language and platform boundaries. UNO objects run and communicate everywhere. UNO reaches this goal by providing the following fundamental framework:

- UNO objects are specified in an abstract meta language, called UNO IDL (UNO Interface Definition Language), which is similar to CORBA IDL or MIDL. From UNO IDL specifications, language dependent header files and libraries can be generated to implement UNO objects in the target language. UNO objects in the form of compiled and bound libraries are called components. Components must support certain base interfaces to be able to run in the UNO environment.

- To instantiate components in a target environment UNO uses a factory concept. This factory is called the service manager. It maintains a database of registered components which are known by their name and can be created by name. The service manager might ask Linux to load and instantiate a shared object written in C++ or it might call upon the local Java VM to instantiate a Java class. This is transparent for the developer, there is no need to care about a component's implementation language. Communication takes place exclusively over interface calls as specified in UNO IDL.

- UNO provides bridges to send method calls and receive return values between processes and between objects written in different implementation languages. The bridges use a special UNO remote protocol (urp) for this purpose which is supported for sockets and pipes. Both ends of the

bridge must be UNO environments, therefore a language-specific UNO runtime environment to connect to another UNO process in any of the supported languages is required. These runtime environments are provided as language bindings.

- Most objects of OpenOffice.org are able to communicate in a UNO environment. The specification for the programmable features of OpenOffice.org is called the OpenOffice.org API.

[...]

Programming with UNO

UNO (pronounced ['ju:nou]) stands for Universal Network Objects and is the base component technology for OpenOffice.org. You can utilize and write components that interact across languages, component technologies, computer platforms, and networks. Currently, UNO is available on Linux, Solaris, and Windows for Java, C++ and OpenOffice.org Basic. As well, UNO is available through the component technology Microsoft COM for many other languages.

UNO is used to access OpenOffice.org, using its Application Programming Interface (API). The OpenOffice.org API is the comprehensive specification that describes the programmable features of OpenOffice.org.

Fields of Application for UNO

You can connect to a local or remote instance of OpenOffice.org from C++, Java and COM/DCOM. C++ and Java Desktop applications, Java servlets, Java Server Pages, JScript and VBScript, and languages, such as Delphi, Visual Basic and many others can use OpenOffice.org to work with Office documents.

It is possible to develop UNO Components in C++ or Java that can be instantiated by the office process and add new capabilities to OpenOffice.org. For example, you can write Chart Add-ins or Calc Add-ins, linguistic extensions, new file filters, database drivers. You can even write complete applications, such as a groupware client.

UNO components, as Java Beans, integrate with Java IDEs (Integrated Development Environment) to give easy access to OpenOffice.org. Currently, a set of such components is under development that will allow editing OpenOffice.org documents in Java Frames.

OpenOffice.org Basic cooperates with UNO, so that UNO programs can be directly written in OpenOffice.org. With this method, you supply your own office solutions and wizards based on an event-driven dialog environment.

The OpenOffice.org database engine and the data aware forms open another wide area of opportunities for database driven solutions

Per esempio, per caricare un documento saranno necessarie svariate chiamate ai servizi di UNO:

```
mxRemoteServiceManager = this.getRemoteServiceManager(unoUrl);
Object desktop = mxRemoteServiceManager.createInstanceWithContext
("com.sun.star.frame.Desktop", mxRemoteContext);
XComponentLoader xComponentLoader = (XcomponentLoader)
UnoRuntime.queryInterface(XComponentLoader.class, desktop);
PropertyValue[] loadProps = new PropertyValue[1];
loadProps[0] = new PropertyValue();
loadProps[0].Name = "AsTemplate";
loadProps[0].Value = new Boolean(true);
xComponentLoader.loadComponentFromURL(loadUrl, "_blank", 0, loadProps);
```

Un utilizzo intensivo di queste primitive porta ad un aumento esponenziale della complessità e mina le possibilità di utilizzo di UNO stesso in ambienti di produzione, caratterizzati da un ridotto time-to-market.

2 Definizione degli Use Cases

2.1 Creazione di un documento

Attore principale: utente di un applicativo

Obiettivi: creare un documento elettronico in un formato manipolabile da altri utenti e stampabile, rappresentante i dati provenienti dall'applicazione che sta utilizzando.

Flusso principale:

1. l'utente accede all'applicazione;
2. l'utente manipola alcuni dati servendosi dell'applicazione;
3. l'utente attiva la funzionalità di reportistica dell'applicazione;
4. l'utente specifica i dati da esportare e le modalità;
5. l'applicazione crea il documento secondo le esigenze dell'utente.

Flussi Alternativi:

1. in un momento qualsiasi del processo l'applicazione genera una situazione di errore:
    1. l'applicazione registra l'errore
    2. l'applicazione effettua le operazioni per ritornare in una situazione normale.

Tecnologie:

1. l'applicazione è in grado di connettersi all'applicazione OpenOffice.org 1.1 o superiore.

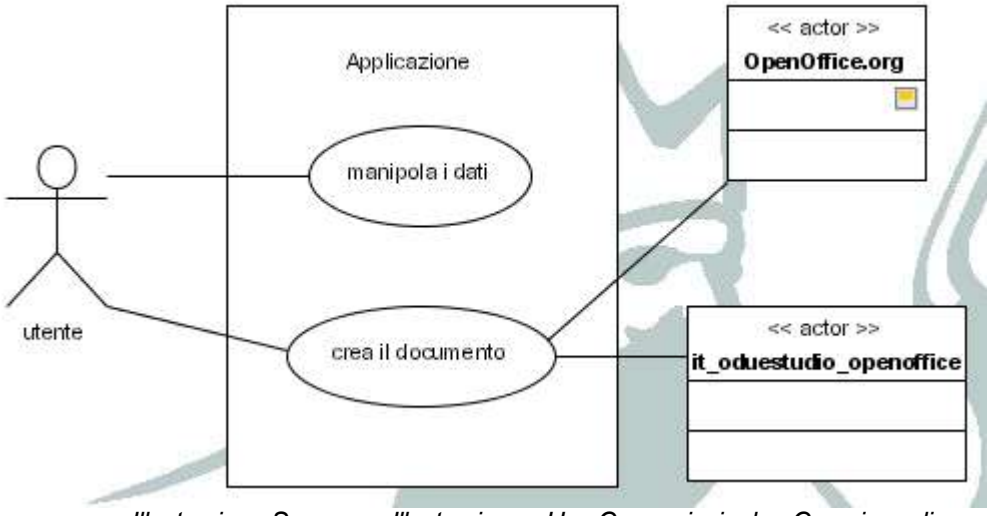

Illustrazione Sequenza Illustrazione - Use Case principale - Creazione di un documento.

3 Analisi dei requisiti

Si definiranno i principali requisiti che le API dovranno soddisfare:

3.1 Requisiti funzionali

1. Sarà possibile utilizzare le API agevolmente in applicazioni Web e GUI.
2. Sarà possibile utilizzare completamente le funzionalità di UNO.
3. Sarà possibile per lo sviluppatore estendere le funzionalità ed inglobarle nelle API in modo omogeneo.
4. Sarà possibile tracciare i processi che avvengono all'interno delle API e definire *callbacks* per gestire particolari eventi.
5. Sarà possibile gestire la connessione ad OpenOffice.org all'interno delle API.
6. Sarà possibile utilizzare le API con processi automatizzati per la gestione di documenti, eventualmente con strumenti grafici o IDE;
7. Sarà possibile per lo sviluppatore tenere traccia delle operazioni effettuate e del loro esito.

4 Domain Model

Da una prima analisi sugli attori e sulla definizione delle responsabilità emergono distinte le necessità di:

- definire una singola unità di lavoro, o **operazione**, assieme alle capacità di eseguire l'operazione stessa – oggetto **`OODocumentOperation`;**
- rappresentare l'oggetto della operazione e le sue caratteristiche – oggetto **`OODocument`;**
- creare i documenti in modalità uniforme e possibilmente automatizzata (per operazioni ripetitive) – oggetto **`OODocumentManager`**;
- rappresentare agevolmente le proprietà di una operazione su un documento – oggetto **`OODocumentProperties`**.

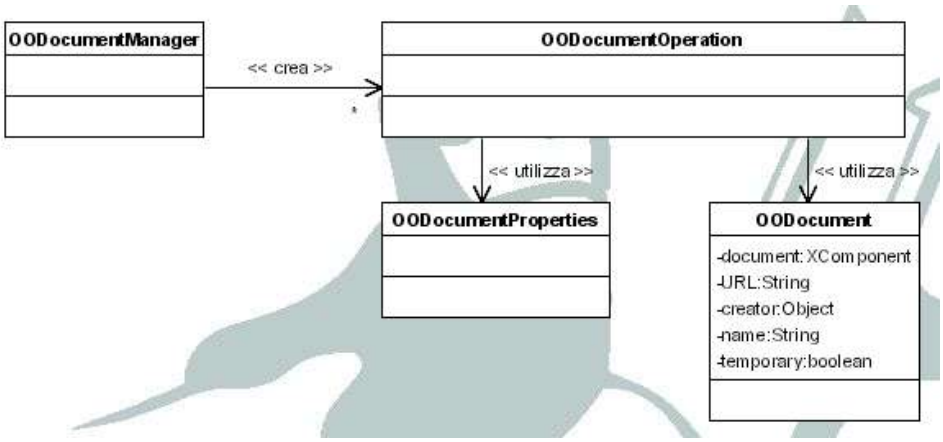

Illustrazione Sequenza Illustrazione - Domain model principale.

L'oggetto OODocumentManager è allo stesso tempo un *factory ed un facade* verso lo sviluppatore, fornendo quindi un agevole accesso alle funzionalità delle API. Inoltre è progettato in modo da potere essere agevolmente utilizzato in ambiti diversi mediante subclassing,

L'utilizzo di tali pattern è giustificato dalla necessità di soddisfare i requisiti 1,3,4,6,7.

E' particolarmente cruciale in questo senso il requisito 3 (estensibilità): lo sviluppatore non si dovrà occupare di come l'operazione verrà recuperata, né di inizializzarla propriamente. Inoltre, come vedremo, questo approccio risulta particolarmente utile per la gestione degli eventi e del logging

5 Sequence Diagrams

La sequenza tipica delle operazioni da eseguire per eseguire una operazione sarà generalmente la seguente:

- creazione di un oggetto OODocument e popolazione delle relative proprietà;
- creazione dell'operazione;
- popolazione delle proprietà dell'operazione;
- esecuzione dell'operazione.

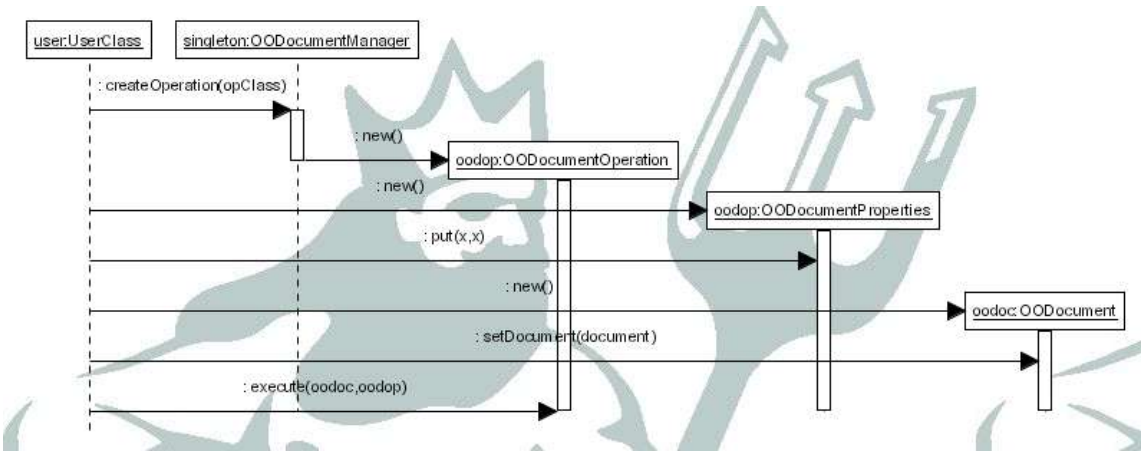

Illustrazione Sequenza Illustrazione - Sequence Diagram dell'utilizzo di una operazione

Per chiarezza, si allega una parte del codice che si vorrà realizzare dal lato utente:

```
OODocument oodoc = new OODocument();

oodoc.setName("test");

OOTemplateLoader gen = (OOTemplateLoader) wm.createOperation
(OOTemplateLoader.class);

OODocumentOperationProperties oodop = new OODocumentOperationProperties
();

oodop.put(OOTemplateLoader.LOAD_URL, config.get
(OODocumentManager.TEMPLATE_FOLDER) + "template.sxw");

gen.execute(oodoc, oodop);
```

6 Class Diagrams

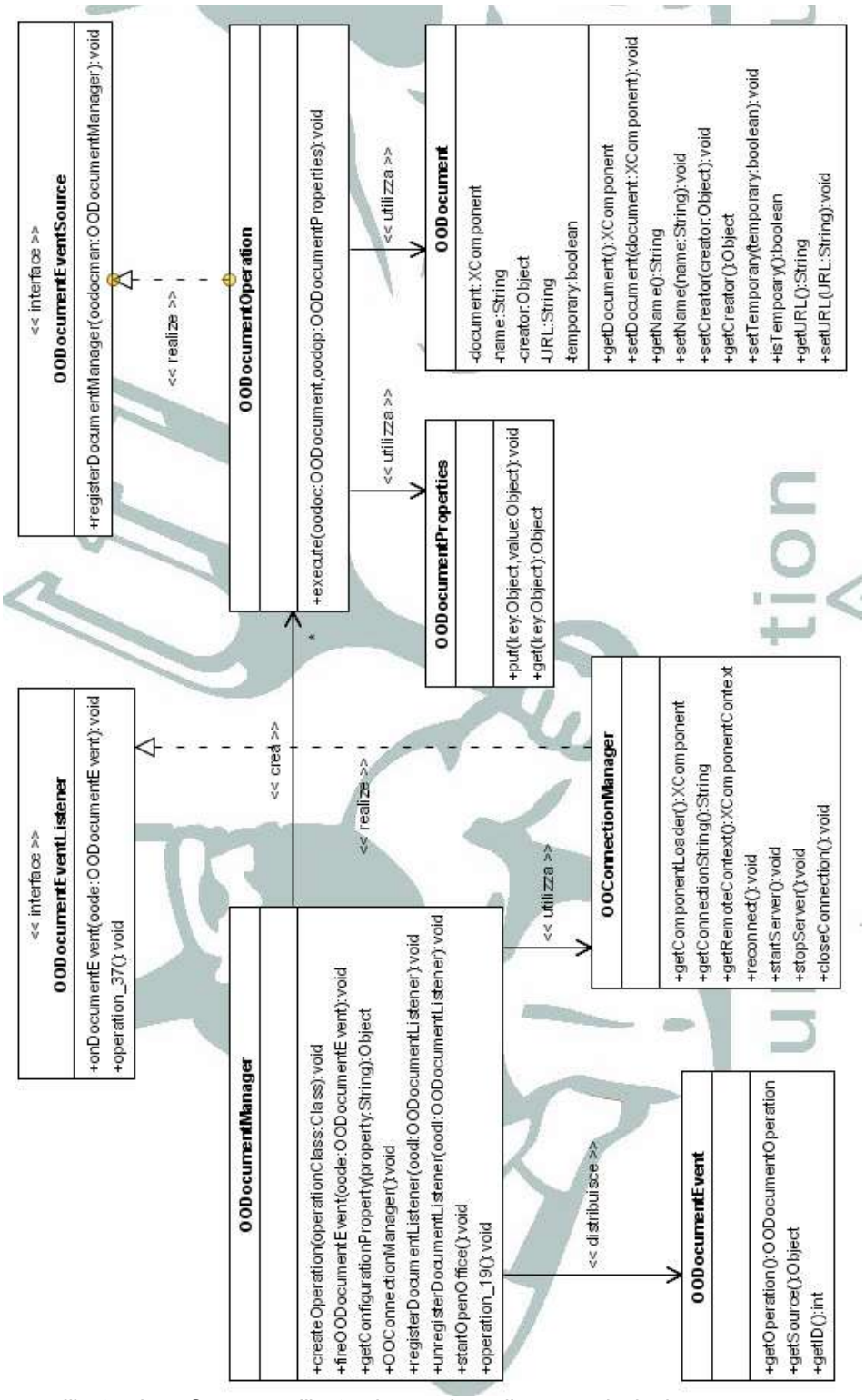

Illustrazione Sequenza Illustrazione - class diagram principale

7 Funzionalità accessorie

7.1 Pipelines: Generatori, Trasformatori, Serializzatori

Il processo di manipolazione di un documento generalmente si compone di tre fasi:

- **Generazione**: si potrà utilizzare un documento vuoto, un documento precedentemente formattato appositamente che funge da *template*, oppure un documento residente sul filesystem. Saranno ammessi tutti i formati gestiti da OpenOffice.org;
- **Trasformazione**: il processo di manipolazione vero e proprio, consistente in una serie di operazioni svolte secondo una determinata sequenza;
- **Serializzazione**: si potrà salvare il documento prodotto in tutti i formati di OpenOffice.org, avvalendosi delle sue ottime funzionalità di conversione di formato.

La suddivisione in queste tre tipologie permetterà ai sottosistemi ed all'utente un maggiore controllo ed un disaccoppiamento più marcato tra le classi. Ad esempio, il sistema di eventi discriminerà il tipo sulla base di queste tre categorie.

Ogni operazione sarà contraddistinta dalle proprietà che ammette, le quali dovranno essere specificate e controllate a runtime. In questo modo sarà possibile riutilizzare lo stesso oggetto `OODocumentProperties` per più operazioni, anche differenti tra loro, ottimizzando risorse e tempo macchina.

Al momento sono disponibili le seguenti operazioni:

OOTemplateLoader

Tipologia: generatore

Descrizione: Carica un documento come template.

Proprietà

Nome	Tipo	Descrizione
LOAD_URL:	String	percorso assoluto del documento da caricare.

OOBookmarkReplacer

Tipologia: trasformatore

Descrizione: inserisce nei bookmarks (segnalibri, nella traduzione italiana di OpenOffice.org) presenti nel documento con i campi specificati. Il bookmark è una 'ancora' e ha la caratteristica di avere un nome univoco all'interno del documento.

Proprietà

Nome	Tipo	Descrizione
BOOKMARKS	OOBookmarks	Dizionario (chiave-valore) dei campi da inserire

OODocumentInserter

Tipologia: trasformatore

Descrizione: inserisce il documento nel bookmark specificato. Se accoppiato ad altre operazioni, fornisce l'equivalente della funzionalità di 'copia ed incolla', non implementata al momento.

Proprietà

Nome	Tipo	Descrizione
SOURCE_DOCUMENT	String	Percorso assoluto nel filesystem del documento da inserire
BOOKMARK_NAME	String	Bookmark indicante il punto in cui inserire

OOSpreadSheetCellReplacer

Tipologia: trasformatore

Descrizione: inserisce nelle celle di un foglio di calcolo i valori indicati.

Proprietà

Nome	Tipo	Descrizione
SPREADSHEET_NAME	String	Nome del foglio di calcolo all'interno del documento
TABLE_CONTENT	OOTable	Tabella delle celle. Se il valore in una cella non è nullo lo sostituisce

OOTableGenerator

Tipologia: trasformatore

Descrizione: inserisce in una tabella il contenuto specificato.

Proprietà

Nome	Tipo	Descrizione
TABLE_NAME	String	Nome della tabella
TABLE_CONTENT	OOTable	Contenuto della tabella
TABLE_ROW_START_INDEX	Integer	La riga da cui iniziare ad inserire il contenuto

OOTableHeaderGenerator

Tipologia: trasformatore

Descrizione: inserisce in un bookmark una tabella vuota

Proprietà

Nome	Tipo	Descrizione
TABLE_NAME	String	Nome della tabella
TABLE_HEADERS	String[]	Nomi degli headers
ROWS	Integer	Numero di righe della tabella
COLS	Integer	Numero di colonne della tabella

Nome	Tipo	Descrizione
BOOKMARK_NAME	String	Punto nel quale creare la tabella

OOTableHeaderGenerator

Tipologia: trasformatore

Descrizione: inserisce in un bookmark una tabella vuota

Proprietà

Nome	Tipo	Descrizione
TABLE_NAME	String	Nome della tabella
TABLE_HEADERS	String[]	Nomi degli headers
ROWS	Integer	Numero di righe della tabella
COLS	Integer	Numero di colonne della tabella
BOOKMARK_NAME	String	Punto nel quale creare la tabella
TABLE_CONTENT	OOTable	Il contenuto della tabella

OOTextFieldReplacer

Tipologia: trasformatore

Descrizione: sostituisce i textfields (campi di testo nella versione italiana) con i valori imposti. I textfields hanno la caratteristica di comparire in diversi punti con lo stesso nome.

Proprietà

Nome	Tipo	Descrizione
TABLE_NAME	String	Nome della tabella
TABLE_HEADERS	String[]	Nomi degli headers
ROWS	Integer	Numero di righe della tabella
COLS	Integer	Numero di colonne della tabella
BOOKMARK_NAME	String	Punto nel quale creare la tabella

OOCalcPDFSerializer

Tipologia: serializzatore

Descrizione: salva il foglio di calcolo come PDF

Proprietà

Nome	Tipo	Descrizione
DESTINATION_FILE	String	Percorso assoluto del documento da salvare

OOPDFSerializer

Tipologia: serializzatore

Descrizione: salva il documento come PDF

Proprietà

Nome	Tipo	Descrizione
DESTINATION_FILE	String	Percorso assoluto del documento da salvare

OOSpreadSheetExcelSerializer

Tipologia: serializzatore

Descrizione: salva il foglio di calcolo come foglio di Excel (.xls)

Proprietà

Nome	Tipo	Descrizione
DESTINATION_FILE	String	Percorso assoluto del documento da salvare

OOWriterPDFSerializer

Tipologia: serializzatore

Descrizione: salva il documento come PDF

Proprietà

Nome	Tipo	Descrizione
DESTINATION_FILE	String	Percorso assoluto del documento da salvare

OOWriterSXWSerializer

Tipologia: serializzatore

Descrizione: salva il documento come documento di OpenOffice.org

Proprietà

Nome	Tipo	Descrizione
DESTINATION_FILE	String	Percorso assoluto del documento da salvare

7.2 Macro

Per agevolare ulteriormente lo sviluppatore, sono state previste alcune macro. L'obiettivo di queste classi è di richiamare automaticamente una serie di operazioni con un metodo unico.

Le specifiche di creazione e di utilizzo di queste classi è libero e lasciato allo sviluppatore. Per l'utilizzo si faccia riferimento alla documentazione in formato javadoc delle API.

BookmarkReplacer

Carica un documento, vi sostituisce i bookmarks, quindi lo salva col nome specificato.

DocumentInsertion

Carica un documento, ve ne inserisce un altro nel bookmark selezionato e lo salva.

TableProducer

Carica un documento, vi crea una tabella con l'operazione OOTableProducer e lo salva.

TextFieldReplacer

Carica un documento, vi sostituisce i textfields, quindi lo salva col nome specificato.

7.3 Gestione Eventi

L'oggetto `OODocumentManager` incorpora le funzionalità di gestione degli eventi. L'implementazione segue la classica metodologia di gestione degli eventi di java: un oggetto che vuole registrarsi come ascoltatore deve implementare l'iterfaccia `OODocumentListener` e chiamare il metodo `registerDocumentListener()` sull'oggetto `OODocumentManager`. Questo, ogniqualvolta verrà richiamato il metodo `fireOODocumentEvent()`, richiamerà il metodo `onDocumentEvent()` sui listener registrati, in modo asincrono e sequenziale, nel thread di esecuzione del metodo `fireOODocumentEvent()`.

L'implementazione delle operazioni attraverso subclassing permette di generare automaticamente due eventi, uno prima dell'esecuzione ed uno dopo l'esecuzione dell'operazione stessa.

Ad esempio, nel caso venisse eseguita l'operazione OOTemplateLoader, sarà lanciato un evento di tipo PRE_DOCUMENT_GENERATED prima dell'esecuzione e un evento di tipo POST_DOCUMENT_GENERATED dopo, anche nel caso che l'operazione non abbia avuto esito positivo.

7.4 Logging

L'intera API è realizzata utilizzando il framework di logging Log4j. Si veda la documentazione di log4j, reperibile da http://jakarta.apache.org per una approfondita discussione sull'utilizzo.

7.5 Gestione delle eccezioni

Per quanto riguarda la gestione delle eccezioni è stato necessario tenere conto delle possibili eccezioni che UNO può generare. Data la sua vastità si è preferito utilizzare il principo della trasparenza, rilanciando cioè le eccezioni generate per poterle manipolare dall'utente. Le eccezioni sono solitamente di tipo `Throwable`, data la possibilità di errori a runtime di UNO; le operazioni lanciano quindi non solo `Exception`, ma `Throwable`.

7.6 Utilizzo in ambiente Web - OOWebDocumentManager

Tale classe è pensata per l'utilizzo in una applicazione web. Estende la classe OODocumentManager, ed è pensata per essere inserita nella sessione utente per potere essere richiamata agevolmente, implementando alcune funzionalità specifiche.

La classe fornisce metodi per aggiungere documenti prodotti dall'utente in una lista interna durante la durata della sessione. Il comportamento di default è di inserire tutti i documenti che non sono temporanei che vengono serializzati. In questo modo l'applicazione potrà mostrare in qualsiasi momento l'intera reportistica generata.

L'oggetto implementa inoltre l'interfaccia HttpSessionBindingListener, che permette all'application server di notificare la classe dell'uscita dell'utente dall'applicazione. Quando tale evento si verifica, vengono eliminati tutti i documenti prodotti nel corso della sessione.

7.7 Utilizzo in ambiente GUI – OOGuiDocumentManager

Nel caso si debba utilizzare le API in una applicazione dotata di interfaccia grafica, l'unico problema che si pone è quello di potere renderne disponibili le funzionalità in ogni punto. Si è reso necessario quindi utilizzare una classe wrapper che, mediante il pattern singleton, implementasse questo requisito.

8 Organizzazione dei packages

Si definiscono i seguenti packages, in base alle responsabilità che questi dovranno tenere:

Nome package	Descrizione
it.oduestudio.openoffice	Fornisce le funzionalità basilari: gestione operazioni, connessione ad OpenOffice.org, gestione eventi ed eccezioni
it.oduestudio.openoffice.generators	Contiene la definizione e le implementazioni delle operazioni di creazione documenti
it.oduestudio.openoffice.transformers	Contiene la definizione e le implementazioni delle operazioni di modifica documenti
it.oduestudio.openoffice.serializers	Contiene la definizione e le implementazioni delle operazioni di salvataggio documenti
it.oduestudio.openoffice.utils	Fornisce funzionalità accessorie e classi di appoggio
it.oduestudio.openoffice.gui	Contiene le classi necessarie all'utilizzo in ambiente GUI
it.oduestudio.openoffice.web	Contiene le classi necessarie all'utilizzo in ambiente Web

9 Esempi di utilizzo

9.1 Inizializzazione e connessione ad openoffice

La connessione ad OpenOffice.org viene effettuata automaticamente durante la istanziazione dell'oggetto `OODocumentManager` e può eventualmente essere richiamata con il metodo `OODocumentManager.getConnectionManager().init()`.

```
Properties config = new Properties();
config.load(getClass().getResourceAsStream("/openoffice.properties"));
config.load(getClass().getResourceAsStream("/log4j.properties"));
OODocumentManager wm = new OODocumentManager(config);
```

9.2 Sostituzione di bookmarks

```
OODocument oodoc = new OODocument();
[...]
OOBookmarks htBmk = new OOBookmarks();
htBmk.put("bookmark","hello, world!");
[...]
OOBookmarkReplacer transformer = (OOBookmarkReplacer)
oodc.createOperation(OOBookmarkReplacer.class);
OODocumentOperationProperties oodop = new OODocumentOperationProperties
();
oodop.put(OOBookmarkReplacer.BOOKMARKS, htBmk);
transformer.execute(oodoc, oodop);
```

9.3 Manipolazione di tabelle

```
OODocument oodoc = new OODocument();
[...]
OOTableGenerator tablegen = (OOTableGenerator) wm.createOperation
(OOTableGenerator.class);
OODocumentOperationProperties oodop = new OODocumentOperationProperties
();
oodop.put(OOTableGenerator.TABLE_NAME, "tabella");
OOTable ooTable = new OOTable(1, 1);
ooTable.insertItemAt("Hello, World!", 0, 0);
oodop.put(OOTableGenerator.TABLE_CONTENT, ooTable);
oodop.put(OOTableGenerator.TABLE_ROW_START_INDEX, new Integer(1));
tablegen.execute(oodoc, oodop);
```

9.4 Estendere le funzionalità

Come scrivere una operazione

Per creare una nuova operazione è necessario:

- Estendere le funzionalità della tipologia di operazione desiderata (AbstractOODocumentGenerator, AbstractOODocumentTransformer, AbstractOODocumentSerializer);
- implementare il relativo metodo astratto.

```
import it.oduestudio.openoffice.OODocument;
import it.oduestudio.openoffice.OODocumentOperationProperties;
import it.oduestudio.openoffice.generators.AbstractOODocumentGenerator;

public class MyGenerator extends AbstractOODocumentGenerator {
    protected void doGenerate(OODocument oodoc,
OODocumentOperationProperties oodop) throws Throwable {
    }
}
```