Describe Functionalities Technical Document

Table of Contents

Purpose

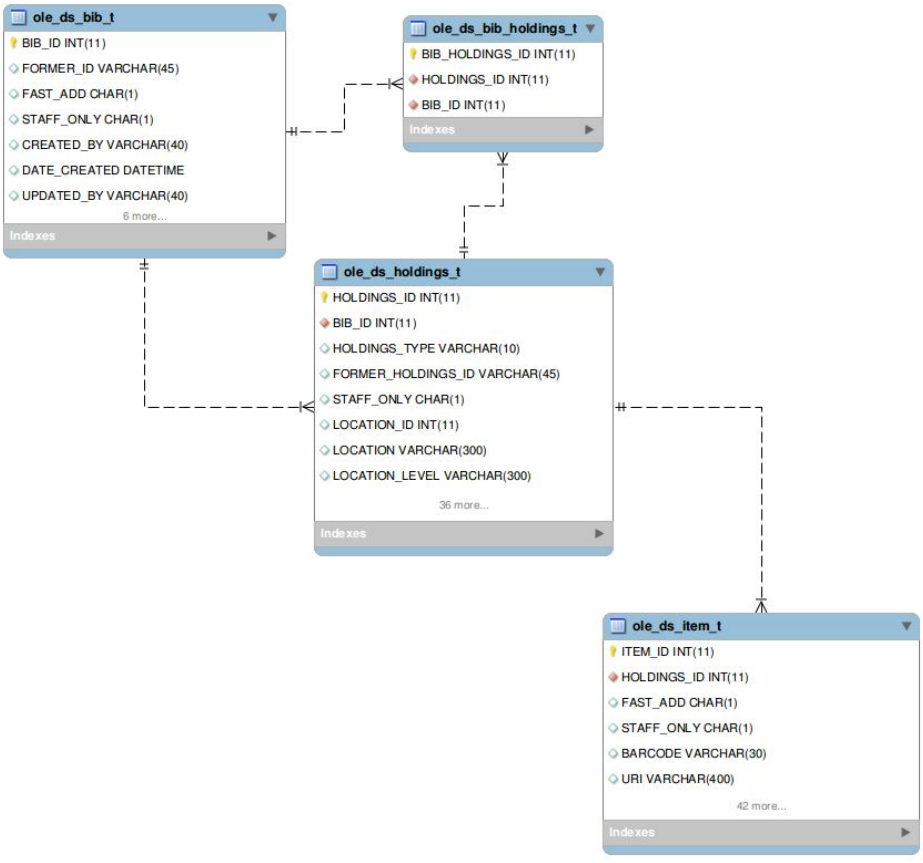
The Describe module of OLE has functionalities to create, update, delete and maintain Bibliographic data. These include ability to create bibliographic records through MARC and Dublin core editors, transferring titles, holding or items amongst bibliographic records, linking monographs to serials and importing a MARC file directly into the system.

This document would detail the logic and working of these functionalities to facilitate easy understanding for future developers and implementers.

Dependencies

Though the bibliographic data resides in Database tables, Solr doesn't make use of data from the database. It neither reads nor writes data to database tables. The indexing is done and the search queries are constructed as Solr queries.

The RDBMS tables are also used. The significant ones are shown below.



Data Model

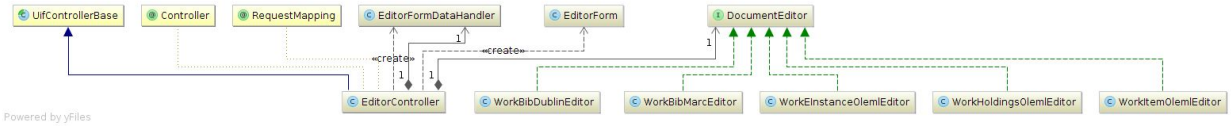
The Solr data model is unique and does not resemble the conventional RDBMS (Relational DataBase Management System) data models used throughout the rest of the system. However, they share some similarities with conventional data models.

More detailed information on Data Model is available here.

Service Interface Design

Editor

OLE provides two different editors depending on the format of the bibliographic record. The two formats supported are Dublin core and Marc.



The *load* method of EditorController is where the loading of the editor page is handled. Depending on the document format (dublin core or marc) and document type (Bib, holding or item) the editor page is presented with a form to the user.

Marc Editor

On submission of the form, the *save* method of the EditorController class is called. The method retrieves various form data and based on which certain manipulations are done. It then calls the *saveDocument* method of the WorkBibMarcEditor class. The *buildDataFieldList* method of the WorkBibMarcEditor class is called where the datafield information are extracted from the form data. Following this the *validateMarcEditorData* method is called which returns a boolean value. Validation is based on the existence of leader, tags and values for both control and data fields. After validation, the bib record is built. Finally the *createBib* method of the DocstoreServiceImpl class is called. The *createBib* method of the DocstoreStorageService class adds a bibliographic data entry in the database. The *createBib* method of the DocstoreIndexService class builds and adds the Solr document into the Solr index.

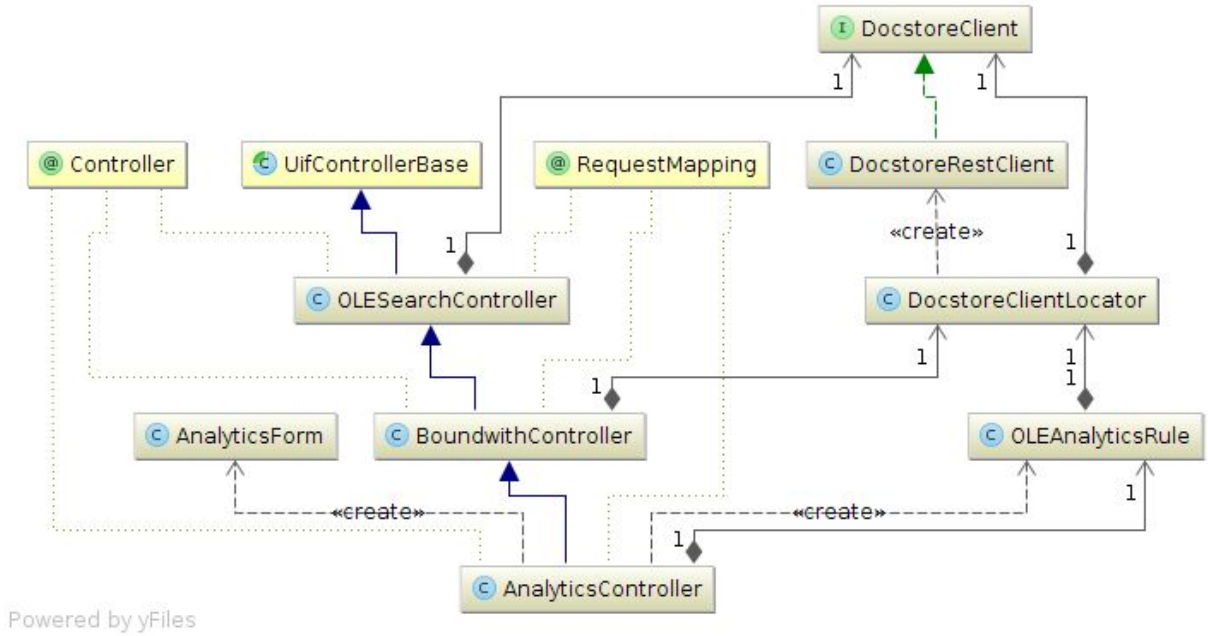
Dublin Core Editor

On submission of the form, the *save* method of the EditorController class is called. However, the DocumentEditor parameter is changed and hence further processing is done in the *saveDocument* method of WorkBibDublinEditor class and not the WorkBibMarcEditor class. All the content is built in this class and saved here.

Analytics

An analytic record for an item is actually a monographic biblio for an item attached to a serial record. Such analytic record tend to be well detailed. This is used for special issues of journals which can also be classified as independent books on their own. They are often assigned both

an ISBN and ISSN/Volume/Issue. It is important for patrons (borrowers) to be able to search for these records both as part of the serial as well as independent, monograph record.



The *start* method of AnalyticsController class is where the loading of the Analytics page is handled. The page allows for a search of bibliographic records from where records can be selected as either the series or the Analytic record.

Once the records are selected and a monograph record in Analytics is selected to create an analytic relation under a particular series through the Create Analytics Relation button, the *createAnalyticsRelation* method of the AnalyticsController is called. The method calls the *validateInputForAnalytics* method of OLEAnalyticsRule class to validate data regarding the analytics relation in the form.

After further validation of the tree structures of both the series and the Analytics record in the *selectCheckedNodesForSeriesTree* and *selectCheckedNodesForAnalyticsTree* methods of OLEAnalyticsRule class, the *addAnalyticRelation* method of AnalyticsController class is called. The *buildIds* method is called from within the *createAnalyticsRelation* method of DocstoreRestClient class. Then the item id and the serial holding id are passed through a REST request through the *postRequest* method of the DocstoreRestClient class and the Analytics relation established.

Bound-With

Bound-With addresses the need to create a single record for bibliographic records of two or more resources which were originally issued as independent resources but have subsequently been placed together under a single cover or packaged together. The libraries would most likely bind those resources which need a lot of cross referencing and are typically checked out together by patrons.

The *start* method of BoundwithController class is responsible for the loading of the Bound-With page in OLE. It uses the BoundwithForm class to collect data pertaining to the Bound-With. The page allows for searching through Bibliographic data and provides option to select records that are to be bound together.

The *performBoundwith* method of the BoundwithController is called when the resources in tree1 and tree2 of the Boundwith screen are bound together. The various data pertaining to the resources in the respective trees are retrieved from the BoundwithForm. The data is passed to the *selectCheckedNodesForTree1* and *selectCheckedNodesForTree2* methods of the BoundwithController class for validation. The *validateInput* method is also used to validate the input parameters. Finally after all the validations, the *performBoundwith* method is called. The *boundWithBibs* method of the DocstoreRestClient class finally uses a REST service to bind together the records.

Loading Your Existing Bounds-Withs Into OLE:

This is the logic we used to load our existing “boundwiths” into OLE (from Sirsi):

Looping through all of your bibs to detect whether or not it is a boundwith:
For each bib -- check for it's id in the bound with spreadsheet (col. A). ...this is based on the sirsi extract of bound-withs

If found:

1. Create a bib
2. Create it's holding record. This will serve as the parent record
3. Create bibs for all of it's related bound-withs from col C of the spreadsheet. These bibs should not have holdings or items.
4. Insert into the bound-with table (ole_ds_bib_holdings_t) an entry for all of the bib ids (including the parent) attaching them to the parents holdings id.

An example we looked at from the attached spreadsheet is "frou frou" -- line 71

23028 serves as the parent bib (in col a) and it has 14 rows in total in the spreadsheet -- so 15 things bound together...

That means
23028 will be a bib with a holding record
985680 bib no holding
985681 bib no holding
985682 bib no holding
985683 bib no holding

985684 bib no holding
985685 bib no holding
985687 bib no holding
985688 bib no holding
985689 bib no holding
985690 bib no holding
985691 bib no holding
985692 bib no holding
985693 bib no holding
985694 bib no holding

and in the ole_ds_bib_holdings_t these will be the 15 entries

holdings_id (i made this up)	bib id
898989898	23028
898989898	985680
898989898	985681
898989898	985682
898989898	985683
898989898	985684
898989898	985685
898989898	985687
898989898	985688
898989898	985689
898989898	985690
898989898	985691
898989898	985692
898989898	985693
898989898	985694

Transfer

Transfer facilitates transferring bib, holding and item records between other bib records. Similar to Bound-With, two different trees can be retrieved and the records transferred between the trees through this functionality.

The *start* method of the TransferController class controls the loading of the Transfer page in OLE. The page has ability to search and assign bib records as two different trees and the ability to move record from one tree to another. Similar to Bound-With, validation is done in the

selectCheckedNodesForItemTransferTree1 and *selectCheckedNodesForItemTransferTree2* methods of the TransferController class. Depending on the record, the *transferHoldings* or the *transferItems* method of the DocstoreClient are called to transfer the records.

Single Record Import

Single Record Import page facilitates the importing of a Marc record. The Marc record with an extension of .mrc is expected to have just one bibliographic record in it. For those with multiple bibliographic records, the Batch process can be used.

The *start* method of the ImportBibController class controls the loading of the page. The page is simplistic with a browse button to pick the location of the Marc file and the load and clear buttons.

The *load* method of the ImportBibController class is called when the single record Marc file is located and and loaded. The file content is retrieved as a Multi part file and is converted to XML in the *getMarcXml* method. The *convert* method of the MarcXMLConverter class is used. The converted XML is updated to include content needed for further processing. The XML is converted to a business object by passing the content to the *fromXML* method of BibMarcRecordProcessor class and added.

Service Interface Design (REST/SOAP)

OLE supports REST services for Solr functions. This is documented here.

User Interface Design

The screens use KRAD's UIF (User Interface Framework). A very good guide on this can be found here.

Data Importing

Not Applicable.

Data Exporting

Not Applicable

Workflow

Not applicable.

System Parameters

No System Parameters are currently used.

Roles and Permissions

Not applicable.