SDN Contribution

Building Custom Applications Using IBOM

Applies to:

This article or whitepaper applies from mySAP CRM 5.1 onwards.

Summary

Internal Business Object Model (IBOM) is the counterpart technology to any external Business object model used to implement the access to the underlying business logic of a BSP or SAP web-based application. Here we take up an Interactive Web Client application that we built for life sciences industry as part of CRM 5.1. The target audiences are the application developers or consultants who wish to build IC web client application in CRM by harnessing the capabilities of IBOM.

Author(s): Sudipta Sarma

Company: SAP Labs India

Created on: 30 May 2006

Author Bio

My name is Sudipta Sarma and I am an employee of SAP Labs India since 3 years. I am currently working in CRM Product and Technology Unit (PTU) for CHM (Life Sciences) Industries.

Table of Contents

A Short Background

The basic idea of the IBOM was developed during the ESA (Enterprise Service Architecture) feasibility study of CRM where it was evaluated how migrate CRM objects to the ESA architecture.

The approach in ESA is having one ESA proxy class for exactly one business object where the whole business logic functionality is covered. That means that the complete data handling for the business object, for all its dependent objects as well as for all associations to other existing business objects is handled in one class, the proxy class.

Here, the idea to build a counterpart of the *external* business object model internally too was conceived. That is to build an internal business object model and to publish the external business object model on top of it. In between the proxy class and the internal business object model there was a need of a generic layer that delegates the calls automatically using business object model Meta information.

This approach has the following **advantages**:

- It is a transparent internal BO model.
- the technical implementation of each node doesn't matter
- the implementation of a node can be re-used in case the access mechanism is the same (e.g. assignments of campaigns & trade promotions)
- enhance able with generic functionality (e.g. CGPL framework used in marketing applications)
- general mapping mechanism between external and internal attributes possible
- abstraction of the complexity of the proxy class interface
- internal BO model is independent from the external BO model as well as the proxy class
- reducing the effort for integrating the application business logic to proxy class
- well-defined implementation framework (each node is implemented the same way)

One **Disadvantage**:

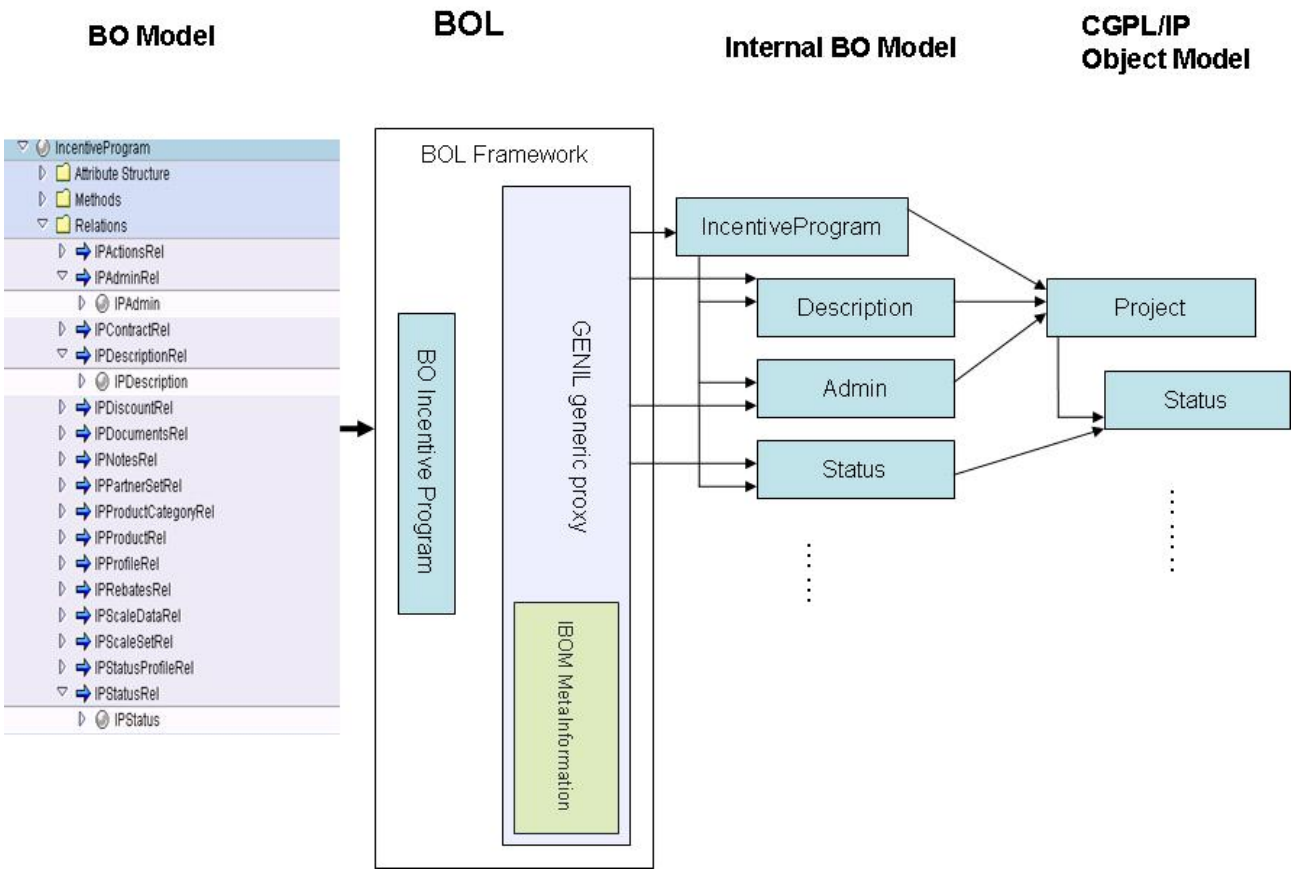
- Overhead for the generic access logic evaluating the business object model Meta information.

Case Study

Since the Internal BO Model approach was designed for all ESA like frameworks, it can easily be applied to BOL (business object layer) to build IC Web Client UI.

The IC web client based application under consideration is **Incentive Program or IP**. The basic application BO modeling is done like any other IC web client application with a root node (IncentiveProgram) and other dependant or child nodes like short texts (IPDescription), status (IPStatus), administrative data (IPAdmin) etc. But the IBOM Meta information needs to be maintained which will define the IBOM nodes, the mappings between the BOL nodes and the IBOM nodes and the different parent-child relationships.

The architecture was conceptualized as in the following diagram.



IBOM Meta Information

In the BOL framework, the publishing of the Meta information is in the responsibility of the application, more precisely it is provided from the proxy class. The meta-data information describes the names of the nodes, its relations, attributes etc.

This is valid for all information we can see in the specific BO model (BOL or ESA). We are having the following mappings for BOL:

Node Name → BOL Node Name

Query Name → BOL Query Name

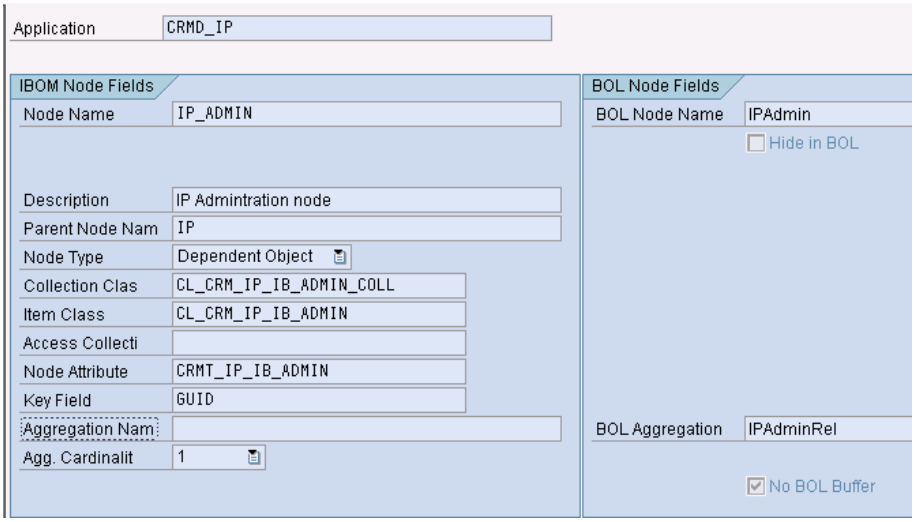
Action Name → BOL Method Name

Aggregation/Composition Name → BOL Aggregation Name

Associations Name → BOL Association Name

The transaction for Meta-information maintenance in case of using BOL is CRMC_MKTIB_IL_CUSTOM

An example of the customizing to be maintained for a IBOM node is as follows:



Collection/Item Pattern

The business object model represents a hierarchy of nodes. Beginning with the root node, some dependent nodes are usually subordinated which can have again sub nodes.

The standard object-oriented approach to build these kinds of hierarchies is the collection/item pattern.

Each node exists of a collection and item implementation. Each item can have several sub collections which have again (sub) items.

The collection class is responsible for fetching the data of several objects and to manage these. Furthermore, new items are created out of the collection. E.g. CL_CRM_IP_IB_ADMIN_COLL

The item class is responsible for maintaining and deleting the data as well as getting the sub collections. E.g. CL_CRM_IP_IB_ADMIN

All the C-Create, R-Read, U – Update and D – Delete operations can be achieved using the collection/item pattern.

The **RootCollection/RootItem** is a kind of header object. It is possible to lock and save only on the Root-Level.

- For your implementation of your root collection class, you need to inherit the IBOM class CL_CRM_MKTIB_ROOT_COLL.
- For your implementation of your root item class, you need to inherit the IBOM class CL_CRM_MKTIB_ROOT_ITEM.

With a **SubCollection/SubItem** it is possible to model a parent/child relation (dependent object).

- For the subcollection, your implementation classes for the subcollection would have to inherit the class CL_CRM_MKTIB_SUB_COLL.
- For the subitem, your implementation classes for the subitem would have to inherit the class CL_CRM_MKTIB_SUB_NODE_ITEM.

Note: All Items and Collections have to be created by the ClassFactory. This class is responsible for the compliance with the customized business model and checks the model, before an object is returned. Objects are requested by their names. The ClassFactory is also responsible to add a reference on the specific metainformation object.

This is how you do it as in the following code snippet (CREATE_ITEM method of the collection class for short description):

```
      CALL METHOD cl_crm_mktib_class_factory=>get_instance
        RECEIVING
          rr_factory = lr_class_factory.

      lv_key = lv_language.

      CALL METHOD lr_class_factory->create_sub_item
        EXPORTING
          ir_parent_collection = me
          iv_key               = lv_key
          is_attributes        = space
          ir_foreign_object    = lr_texts
        RECEIVING
          rr_sub_item          = lr_item.

    rr_item = lr_item.
```

Modeling relationships

Both association and aggregation relationships can be modeled in IBOM.

Generic Services

There are a few functionalities available which are exposed as generic services that application developers can re-use. To name a few are Business Partner, Products. This can reduce application development time and effort drastically.

Other nice features

All functionality that is exposed by the IC Web client specific BOL is also provided by IBOM. These include:

- Query handling – provides advanced search functionality. You need to implement the methods of class CL_CRM_MKTIB_QUERY_HANDLER
- Action handling – To handle actions (BOL methods). You need to implement the execute method of class CL_CRM_MKTIB_ACTION_HANDLER.
- Message handling – for the purpose of logging messages for your application. To achieve this, you need to implement the interface IF_CRM_MKTIB_MSG_HANDLER
- Exception handling – IBOM provides the possibility to throw exceptions from the proxy layer and thereby provide additional debugging information. Also, the exception

CX_CRM_MKTIB_APPLICATION_IMPL is intended for Application Developers who are using the IBOM Framework.

- Caching – The IBOM layer does not cache references to collection and item objects as the underlying API layer would mostly already cache the application data. So, there is no global object cache; but a session and an application cache are made available for the action handler and for association relationships respectively.

Generic GenIL Proxy Layer

To integrate the specific Business Object Model implementation in the BOL for IC Web Client, the Application developer has to derive from the Interaction layer (IL) Proxy Class. Because this Class is derived from the GENIL_ABSTR_COMPONENT2 from the BOL Library it is possible to add it as an interaction Layer class. You have to redefine only two methods.

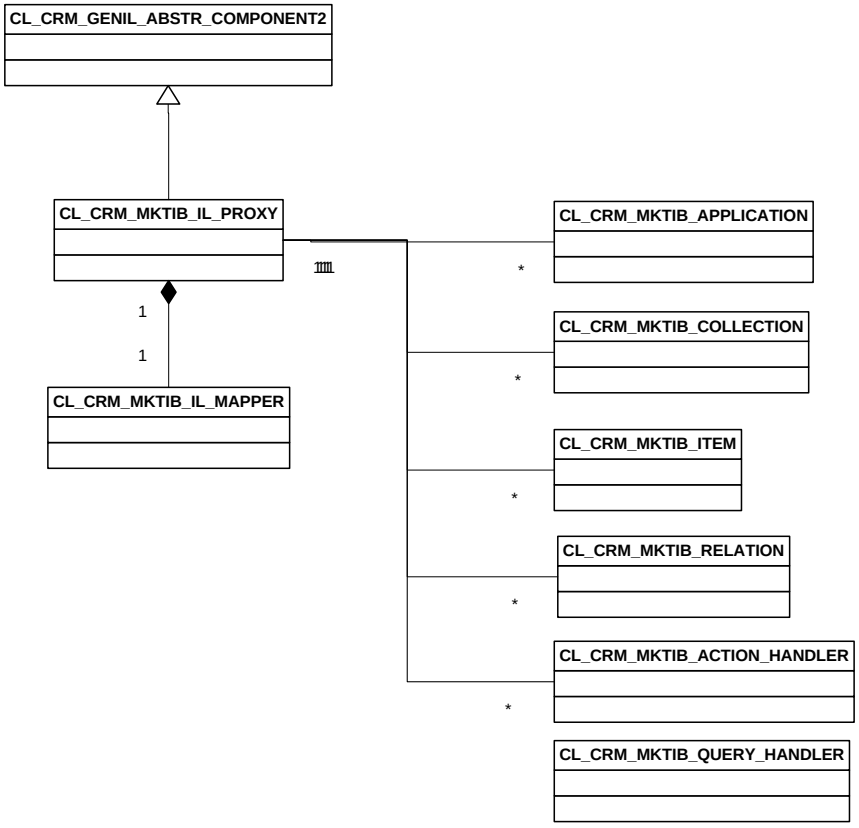
GET_APPLICATION_NAME which should return the name of the application from the object model and

GET_ROOT_NODE_NAME which should return a root node name from the object model

The rest is all taken care of by the framework.

The IL Proxy creates an instance of the specific application for each request from the BOL (stateless), navigates to the requested objects and calls the appropriate functions.

The IL Mapper provides functionality to map external model names to internal model names and vice versa.



This completes your BOL implementation. You can then maintain your BOL customizing for your application component and component set. As a further step, once you test your application in your BOL browser, you can start building your IC Web Client UI.

IBOM Packages

CRM_MKTIB - This package contains classes for building your own component model like Application, Collections and Items, Relations, Metainformation, the ClassFactory, Query Handler and Action Handler.

CRM_MKTIB_IL – Contains the GenIL proxy and the mapping classes. This enables BOL to communicate with the Business Object Model classes.

For the implementation of your specific object model IBOM gurus also recommends you to create two packages. One package for the specific business model implementation, which is dependent from the CRM_MKTIB Package and one package for the specific GENIL proxy implementation dependent from the CRM_MKTIB_IL package.

Conclusion

All in all, the internal BO model or IBOM is developed with a lot of flexibility which enables application developers like me to bring up IC web client or future Web dynpro based applications in no time and with minimal effort. I would rate this technology 9 out of 10.

Happy Developing IBOM applications!

Related Content

1. Twiki Page for IBOM (For internal SAP employees)

Copyright