Increasing Productivity through a Better Management of Business Objects in Economic Applications

Dănuţ-Octavian SIMION
Lumina – The University of South-East Europe, Bucharest, Romania
danut.simion@lumina.org

Abstract

The paper presents the advantages of a better management regarding business objects in economical applications that are used in the production activity. These business objects are usually Enterprise Java Objects that represents clients, orders, resources, data access objects and services used in the logic of the economic applications. The economical flows are very important within these kind of applications because are dependent of each other so a good management leads at an increase of productivity and the profit. These business objects are located in the Business Tier that provides inputs for other tiers and so it is important that the design and implementation are done by a management plan according to the business logic. The Business Tier is represented by business objects and services that allow to store data and to define the business logic of applications. Also the Business Tier is responsible to process client's requests and delivers responses according with the business logic and so his design and implementation is important.

Keywords: *Economic applications, Management process, Business Tier, Java objects integration, Business objects, UML diagram*

JEL Classification: O12

Introduction

The management activity involves a good organization of business flows that are implemented in economical applications. In these applications The Business Tier manages all the business objects that represent entities like products, customers, orders, data access objects and services used in the logic of the economical applications. A better management of these objects relies on a good design and a rigorous implementation through Java which is a oriented object programming language that is independent from operating systems platforms and can integrate business objects in various economical applications. Java components for Business Tier include different parts like business objects that map entities according to the specific of each enterprise. The Java programming language can separate the main tiers for better and easy ways of building business applications. The main tiers of a business application are Presentation Tier, Business Tier and Integration Tier. Separating these tiers can dissociate the working processes of the programmers and web designers, and so the business logic of the applications can be separated from the design work and different parts of applications could be integrated more easily and the duration of completing these tasks is shorter [Doug Lea, 2010], [James W. Cooper, 2008]. The

Business Tier is referring to client's access at different components that are structured in java classes that are customized for each type of business applications and contains description for actions like orders, supply, purchasing, delivery and other economic activities.

1. Model for Business Tier

Web applications for business help business development for companies that adopt these technologies in companies that are in various formats can be used in an integrated way. Current technologies are oriented data types existing within companies, including those older technologies, DBMS from previous generation, unstructured data, old applications which do not meet current standards. The main contributions of IT are finding new ways to access different types of structured data in databases and presenting various ways to develop Web applications using JAVA technology. These new types of applications lead to the development of companies by allowing several users to access shared within their different collections of data, but are possible and open for business Internet users. For building Web applications, the first step is to identify how to structure data: structured, semi-structured or unstructured. Structured data found in databases are managed by DBMS's, semi-structured data files are in XML files and unstructured data that are in the Office file types. Very important for the business is when the management of the company knows the amount of data that is present in companies and to inventory main compartments that produce documents and to establish key information flows. In general Web applications enable companies to open the web world, or build useful applications within companies, especially in the exchange of data between departments. If the data is structured in databases, this makes it possible to transfer them to the DBMS performance sites, thus centralizing data in the various compartments. Semi-structured data existing in XML files, involve lower costs for firms and greater flexibility in their management and creates the possibility of building new Web applications with open-source technologies (programming languages such as Java, PHP, Perl, etc.). The data in these files are easily updated, edited, interrogated and especially easy to store in a file system present on servers that have the operating system like Linux [Doug Lea, 2010], [James W. Cooper, 2008], [SAP, 2012].

Important elements in Web applications are data formats, data models, and ways of transforming XML documents into Java classes, managing XML Web services for SOA (Service Oriented Architecture), and also using the JavaScript language in Web. Web instruments are for managing pages and Servlets, JSP's (Java Server Pages), JSF's (Java Server Faces), JSTL's (Java Server Pages Tag library). There are core facilities such as libraries, libraries functions, data formatting, and activities like processing XML files and databases, working with Struts (Framework open source JSP / Servlet), Spring (application Framework for assembling components of an application via configuration files) and Hibernate (Java library for ORM - object-relational mapping): dependency injection, aspect oriented programming; mapping files, sessions and transactions, operations CRUD (create, read, update and delete) query language HQL (Hibernate Query Language) that is the basis for Web applications that enable interaction between users and databases. Connecting users to the database would not be possible without implementing connectivity drivers that are specific to each DBMS's, included in this technology that is very complex [Binildas A. Christudas, 2011], [David Geary, 2010], [Doug Lea, 2010]. Among the best performing drivers are implemented in Java, which offers multiple amenities and

possibilities of customization required by different types of Web applications. An important role in these applications have the Java beans that reflects the database entities, conduct transactions, user commands running their use in technologies such as Java RMI (Java Remote Method Invocation - is an API (Application Programming Interface) actions performed equivalent remote procedure calls), CORBA (Common Object Requesting Broker Architecture - is a standard defined by the Object Management Group that enables software components written in different languages to run on different machines that work together). Java Beans allow separation of the user interface from the application logic (the unseen - the business model) and thus allow unbundling by web designers programmer's activities. There are ways to access data through Web services: standards RPC (Remote Procedure Call) and XML-RPC (creating a client XML RPC client creating a server with XML RPC server), SOAP (Simple Object Access Protocol - an XML-based protocol), PHP ADODB library (database abstraction library for PHP, based on the concept of ActiveX Data Objects Microsoft). Analyzing C # for Microsoft SQL Server, XML, HTML can create HTML detailed aspects of Transact-SQL and make web task, XML schemas, transformations XSLT (Extensible Style sheet Language Transformations), Document Object Model, HTTP queries, NET Technology, extended stored procedures (Open Data Services), transact- SQL commands DBCC (Database Console commands for Microsoft SQL Server) undocumented information schema views. For designing Web applications are drawn relational schemas, diagrams, detailed documentation, technical specifications, etc. that can be organized and structured in UML (Unified Modeling Language) - a universal language that creates software design patterns in object use by software developers and by UBL (Universal Business language) - universal language of business which includes XML libraries that shape specific business documents (invoices, orders, etc.) [David Gallardo, 2009], [Doug Lea, 2010]. An important role in building web applications has XML files.

By adding semantic constraints, application languages can be implemented to build the XML. In particular, XML is used as the specification language for other languages for building Web applications. There are traditional techniques for processing XML files, such as the use of programming languages and APIs SAX (Simple API for XML - a Java API) programming languages and APIs DOM (Document Object Model - a component API of the Java API for XML) transformation and filtering engines use. Recent techniques for processing XML files are: Pull Parsing, Non-Extractive Parsing and Data Binding. SAX is a lexical interface, event-driven, in which a document is read in the way content is serialized and reported as "callbacks" to the handler object, is used by the user design. If the XML data is structured, via Java, this language can convert them into Java classes. Generating classes is made based on validation rules via a compiler XS MARSHALLING, UNMARSHALLING, so that the data present in Java classes can easily be used in Web applications and for activities like display, modify, delete and add data. To use XML files in web applications means important work of transforming XML files. As the transformation engine and filters are: the XSL (Extensible Style sheet Language) that can transform XML files for display and printing. XSL-FO (XSL Formatting Objects) is a declarative language for XML-based page-layout. XSL-FO processor can be used to convert an XSL-FO document in an XML document that does not, for example PDF. XSLT (Extensible Style sheet Language Transformations) is a declarative language for transforming XML documents. An XSLT processor can use an XSLT style sheet to convert the template data tree represented by an XML document into another tree that can be serialized as XML, HTML, text or any other format supported by the processor.

XQuery is a W3C language query construction and processing XML data. XPATH is a tree-like DOM (Document Object Model) data models and path expression, language used for selecting data within XML documents for XSL FO, XSLT and XQuery using XPATH. XPATH function library should include a selection of XML data [James W. Cooper, 2008], [SAP, 2012]. The management of applications focused on important business systems such as ERP (Enterprise Resource Planning), CRM (Customer Relationship Management), WCMS (Web Content Management System), B2B (Business-to-Business) and other business applications.

To illustrate a Business model it can be used the following example:

Implementing a Business model.

```
public class Ex1 {
    // Create a session
private BO_Resource_Session session;
   // Class for business object
private static final Class cls1 =
     BO_Resource_Session.class;
   // Constructor for Ex1.
public Ex1() throws BO_Resource_Exception {
     try {
BO_Resource_Session obj_home =
        (BO_Resource_Session_Home) BO_Service_Locator.getInstance().
          getRemoteHome("Resource", cls1);
       BO_session = obj_home.create();
     } catch (BO_Service_Locator_Exception ex) {
       // Business Object Service Locator exception for
       // application exception
       throw new BO_Resource_Exception();
     } catch (BO_Create_Exception ex) {
       throw new BO_Resource_Exception();
     } catch (BO_Remote_Exception ex) {
       throw new BO_Resource_Exception();
     }
   }
   // Constructor for Business Object id
public Ex1(String id)
throws BO_Resource_Exception {
     // connect to the Business Object bean for id parameter
     reconnect(id);
   }
   // Parameter out - ID client
   // to reconnect to the Business object bean
public String get_BO_ID() {
     try {
return BO_Service_Locator.get_BO_Id(session);
     } catch (Exception e) {
       // Throw exception
     }
```

```
    }
   // Business object method to reconnect using given ID
    public void reconnect_BO(String id) throws BO_Resource_Exception {
     try {
      session =
       (BO_Resource_Session) BO_Service_Locator.getBO_Service(id);
     } catch (BO_Remote_Exception ex) {
      // Business Object Remote exception
      // for application exception
      throw new BO_Resource_Exception();
     }
    }
    // Create a session for the business application
    public BO_Resource_Template_Obj setBO_Current_Resource(String resource_BO_Id)
    throws BO_Resource_Exception {
     try {
      return session.setBO_Current_Resource(BO_resource_Id);
     } catch (BO_Remote_Exception ex) {
      // Business obj service exception for
      // application
      throw new BO_Resource_Exception();
     }
    }
    public BO_Resource_Template_Obj getBO_Resource_Details()
    throws BO_Resource_Exception {
try {
      return session.getBO_Resource_Details();
     } catch (BO_Remote_Exception ex) {
      // Business obj service exception for
      // application
      throw new BO_Resource_Exception();
     }
    }
public void setBO_Resource_Details(BO_Resource_Template_Obj to)
    throws ResourceException {
     try {
      session.setBO_Resource_Details(to);
     } catch (BO_Remote_Exception ex) {
      throw new BO_Resource_Exception();
     }
   }
public void addBO_New_Resource(BO_Resource_TemplateObj to)
   throws BO_Resource_Exception {
    try {
     session.addBO_Resource(to);
    } catch (BO_Remote_Exception ex) {
     throw new BO_Resource_Exception();
    }
```

```
  }
}
```

Remote Interface

```
public interface BO_Resource_Session extends BO_EJB_Object {
public BO_Resource_Template_Obj set_Current_Resource(String resourceBO_Id)
throws BO_Remote_Exception, BO_Resource_Exception;
public BO_Resource_Template_Obj getBO_Resource_Details()
throws BO_Remote_Exception, BO_Resource_Exception;
public void setBO_Resource_Details(BO_Resource_Template_Obj resource)
throws bo_Remote_Exception, BO_Resource_Exception;
public void addBO_Resource(BO_Resource_Template_Obj resource)
throws BO_Remote_Exception, BO_Resource_Exception;
public void removeBO_Resource()
throws BO_Remote_Exception, BO_Resource_Exception;
// Business objects methods
public void addBO_Blockout_Time(Collection BO_blockout_Time)
throws BO_Remote_Exception, BO_Blockout_Time_Exception;
public void updateBO_Blockout_Time(Collection BO_blockout_Time)
throws BO_Remote_Exception, BO_Blockout_Time_Exception;
public void removeBO_Blockout_Time(Collection BO_blockout_Time)
throws BO_Remote_Exception, BO_Blockout_Time_Exception;
public void removeBO_Blockout_Time()
throws BO_Remote_Exception, BO_Blockout_Time_Exception;
// other skill methods
public void addBO_Skill_Sets(Collection BO_skill_Set)
throws BO_Remote_Exception, BO_Skill_Set_Exception;
public void BO_update_Skill_Sets(Collection BO_skill_Set)
throws BO_Remote_Exception, BO_Skill_Set_Exception;
public void BO_remove_Skill_Set(Collection BO_skill_Set)
throws BO_Remote_Exception, BO_Skill_Set_Exception;
 }
```

2. Business Object

When there is little or no business logic in a business operation, applications will typically let clients directly access business data in the data store. A presentation tier component such as a command helper or JSP view, or a business-tier component can directly access a Data Access Object. In this case, there is no notion of an object model in the business tier. The application requirements are fulfilled by a procedural implementation [Binildas A. Christudas, 2011], [James W. Cooper, 2008]. This approach is acceptable for some applications when the data model closely represents the conceptual domain model. If there is a conceptual model that exhibits a variety of business behavior and relationships, implementing such applications using a procedural approach causes the following problems:

- reusability is reduced and business logic code is duplicated;
- procedure implementations becomes lengthy and complex;
- poor maintainability due to duplication and because business logic is spread over different modules.

Business Objects encapsulate and manage business data, behavior and persistence. Business Objects help separate persistence logic from business logic. Business Objects maintain the core business data, and implement the behavior that is common to the entire application or domain. In an application that uses Business Objects, the client interacts with the Business Objects, which manage their own persistence using one of the several persistence strategies. Business Objects implement a reusable layer of business entities that describe the business domain. A Business Object implements a well-defined business domain concept and includes business logic and business rules that apply to that domain concept. Higher-level of business logic that operates on several Business Objects is implemented in a service layer, using Application Service and Session Façade, to isolate the object model from clients, preventing direct access.

Implementing a Business Object

```
public class Ex2 {
private BOCustomer_Data customerBO_Data;

// Contact Info BO is a dependent Business Object
private ContactInfo_BO contactInfo_BO;

public CustomerBO(Customer_Data customer_Data) {
// validate Customer Data values
        this.customer_Data = customer_Data;
    }

public ContactInfoBO getContactInfoBO () {
        if (contactInfoBO == null)
           contactInfoBO = new ContactInfoBO(
           customerData.getContactInfoData());
        return contactInfoBO;
    }}
Business Object - ContactInfoBO
  public class ContactInfoBO {
     private ContactInfoData contactInfoData;

     public ContactInfoBO(ContactInfoData contactInfoData) {
        this.contactInfoData = contactInfoData;
     }
     public BOAddress_Data getBOAddress_Data () {
        return contactInfoBO_Data.getBOAddress_Data();
     }}
```

3. Conclusions

The management activity for economic applications involves a good understanding of the business flows and a rigorous implementation of the business objects according to the specifications. The Business Tier provides the java objects for the Integration Tier and represents the results from the clients requests that are also processed according with the business logic of the applications [Binildas A. Christudas, 2011], [Doug Lea, 2010]. The

objects for business and the business flows are the most representative components of the Business Tier and they work together on a Java platform through classes, interfaces, services and data types definition, so the design and implementation of those is very important [David Geary, 2010], [James W. Cooper, 2008]. The main benefit in using the proper design and implementation is the dissociation between the Presentation Tier, Business Tier and the Integration Tier. The Java programming language improves the ways of building new business applications and to reuse old resources from previous applications thus is easy to answer at the specific demands from the clients. Java applications are more adaptive and robust if it is built on a business logic that implements the basic design and implementation according with the UML diagrams.

References

Binildas A. Christudas, (2011) "*Service Oriented Java Business Integration*", Packt Publishing

David Gallardo, (2009) "*Java design patterns 101*", ibm.com/developerWorks

David Geary, (2010) "*Java Design Patterns*", www.javaworld.com

Doug Lea, (2010) "*Concurrent programming in Java design principles and patterns*", Addison Wesley,

James W. Cooper, (2008) "*Java Design Patterns at a Glance*", www.javacamp.org /designPattern/

SAP Co-Innovation Lab, (2012) "*Evaluating Selected Java Best Practices for Sap Business objects Business Intelligence 4 on V sphere*", SAP Corporation

URI: http://javaworld.com/javaworld/

URI: http://www.packtpub.com/service-oriented-java-business-integration/

URI: http://www.sun.com/software/javaenterprisesystem/

URI: http://www.manageability.org/