A MVC-Based System Architecture for E-Business Applications

Luo Cheng   Lai Mingyong   Zhang Hanjiang   Luo Duanhong
School of Economics and Trade
Hunan University, changansha hunan, 410079

Abstract. This paper discusses the architectures for E-Business application systems. A seven-tier architecture based on MVC (Model-View-Controller) paradigm is proposed for E-Business application systems. We analyzed business process models and built a model frameworks for programming templates. We also provide system developers with a development supporting framework and assisting tools. System developers can rapidly build all layers of source codes for business functions, by using the design results of data models and business models extracted from the framework.

Keywords. E-Business applications, MVC paradigm, Development supporting framework

1. Introduction
The E-Business uses digital and electronic means for commercial data exchange, and facilitates business and commercial activities. Currently there are many E-Business application systems, and for example, Escalade is a set of ERP (Enterprise Resource Planning) software for E-Business and selling enterprises. It uses popular business flows, and satisfies most business requirements by providing a complete set of flexible business functions including subsidiary management modules of clients, selling and buying orders, delivering, storage, classified selling and buying records, and systems. Another example is the enterprise E-Business system POWER EC developed by Powerise, which provides information management software for Chinese middle and small sized enterprises. Various sub-systems for E-Business, ERP, office documents management and mobile office services are seamlessly integrated, which facilitates conveniently data exchange and sharing among sub-systems and well supports the legacy systems in the enterprise. However, as E-Business systems often employ different platforms, application systems, and database management systems, it exhibits a great degree of system heterogeneity and a low degree of software reusability. The traditional software developing procedures and manual productive manners lead to the low development efficiency and product quality.

2. MVC Paradigm
Current E-Business application systems are mostly based on the traditional three-tier (i.e. data tier, logic tier, and presentation tier) architecture popular in Web services, as illustrated in Figure 1.

![Figure 1 Three-tier architecture](image)
It shows the functionalities of every tier together with a corresponding realizing technology. The presentation tier uses JSP, which calls Java Beans to process business logic. The Java Beans in turn calls EJB, which speaks with databases. The advantages include clear function separation, reusability of Beans and EJB, and little interference due to data source changes or web page adjustments. For example, if there is a change in database design and EJB needs to be changed accordingly, JSP and Beans can remain unaltered, as long as the defined interfaces are not changed.

In the MVC paradigm the user input, the modeling of the external world, and the visual feedback to the user are explicitly separated and handled by three types of object, each specialized for its task. The view manages the graphical and/or textual output to the portion of the bitmapped display that is allocated to its application. The controller interprets the mouse and keyboard inputs from the user, commanding the model and/or the view to change as appropriate. Finally, the model manages the behavior and data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state (usually from the controller). The formal separation of these three tasks is an important notion to provide a flexible and powerful system. The advantages of MVC are clear design, high modularity, easy extensibility, and distributed and powerful user interfaces.

3. Seven-Tier Architecture

Considering the aforementioned advantages of MVC, we propose in this paper a system architecture based on MVC for E-Business Applications. We divide an E-Business application system into a seven-tier architecture: data tier, data access tier, data entity tier, business entity tier, business service tier, interface component tier, and controller and interface generation tier, as shown in Figure 2. According to the analysis of E-Business application systems, the common development components include classified commodity management, commodity directory management, commodity information management, commodity classified index, online purchasing, commodity selling statistics, shopping cart functionality, cashier functionality, online payment, commodity display, etc. The commodity information management mainly accesses the data tier and the business tier, whereas shopping cart functionality usually talks with the interface tier.

3.1 Data Tier

The data tier includes all data resources in the form of independent systems, such as database management systems, email systems, multimedia information systems, etc. Data tier is the physical representation of business data and system meta-data (including technical meta-data and business meta-data). When business systems are being designed, standard design must be obeyed to regulate business data model definitions.

3.2 Data Access Tier

Data Access Tier offers an integrated and universal data view to the Data Entity Tier by shielding the details of distributed heterogeneous data resources to exactly integrate the data resources all over the networks. Data Access Tier is mainly designed to solve the problems related to enterprise applications such as database connection, result set, balance, fault tolerance, transaction and security etc. Data Access Tier provides a transparent access to data by hiding the differences among the databases from different vendors.
3.3 Data Entity Tier

Data Entity Tier is consisting of entity objects from applications. Entity object implements the object/relation mapping (i.e. O/R Mapping) so that the upper tiers can access records as objects. Data Entity Tier uses the SQL language.

3.4 Business Entity Tier

Business Entity Tier is based on the Data Entity Tier and offers objects access of the data according to the need of business logics. It stores the structure of the application data. There are two kinds of business entity objects according to object properties. One kind of them is behavior entity objects and the other kind of objects is information entity objects. Behavior entity objects can perform some functions whereas information entity objects only contain information.
3.5 Business Service Tier

Business Service Tier is based on the Business Entity Tier and is composed of business service objects. Business Service Tier implements the business logics of applications. Sometimes business service objects are the behavior entity objects in Business Entity Tier. Business service objects respond to the requests from users by providing suitable services and returning the results to the users. Business service tier takes more care of the relations of business entity than the implementation. It will implement the business functions more complicated through the use of all the objects in Business Service Tier and Business Entity Tier.

3.6 Interface Component Tier

The Interface Component Tier provides the functionalities of all kinds of presentation controls and component models on the entire system interface layout, and is responsible for generating components such as business entity querying, business entity management browsing, business entity editing, functionality control command, etc. According to the design of the business model, business compiler is able to generate the interface component classes based on the development requirements.

3.7 Controller and Interface Generation Tier

Controller is responsible for controlling the submission of interface interactions. It accepts user operation requests, notifies the service objects in the business service tier to process the business requests, and returns the interface content according to relevant meta-data and business processing results. Interface Generation Tier provides various types of basic business interface layout models, including browsing type models, editing type models, and content displaying type models. According to the design of the business model, business compiler is able to build the interface generation classes based on the development requirements.

4. Conclusion

This paper proposes the system architecture based on MVC for E-Business applications. During application system development, the descriptions of data models and business models are extracted from the design process and stored. The descriptions and programming component templates can be used to rapidly generate all kinds of code segments for concrete businesses and facilitate client-oriented development. E-Business system developers can quickly build E-Business applications according to existing business modes, and focus on analyzing and decomposing new complex business flows. They can also extend and accumulate new business modes based upon this architecture, therefore expanding the application area of the platform. The proposed MVC-based architecture can be used not only for developing E-Business application systems, but also for other management information systems.

References

[3] Steve Burbeck, How to use Model-View-Controller (MVC). Available at
