University of Nebraska at Omaha
DigitalCommons@UNO

Student Work

5-2003

Implementation E-commerce application using Lotus Domino

Jianrong Dai
University of Nebraska at Omaha

Follow this and additional works at: https://digitalcommons.unomaha.edu/studentwork

 Part of the Computer Sciences Commons

Recommended Citation

Dai, Jianrong, "Implementation E-commerce application using Lotus Domino" (2003). *Student Work*. 3076.
https://digitalcommons.unomaha.edu/studentwork/3076

This Thesis is brought to you for free and open access by DigitalCommons@UNO. It has been accepted for inclusion in Student Work by an authorized administrator of DigitalCommons@UNO. For more information, please contact unodigitalcommons@unomaha.edu.

IMPLEMENTATION E-COMMERCE APPLICATION USING LOTUS DOMINO

A Thesis-Equivalent Project

Presented to the

Department of Computer Science

and the

Faculty of the Graduate College

University of Nebraska

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

University of Nebraska at Omaha

By

Jianrong Dai

May 2003

UMI Number: EP73450

All rights reserved

INFORMATION TO ALL USERS
The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript
and there are missing pages, these will be noted. Also, if material had to be removed,
a note will indicate the deletion.

UMI EP73450

Published by ProQuest LLC (2015). Copyright in the Dissertation held by the Author.

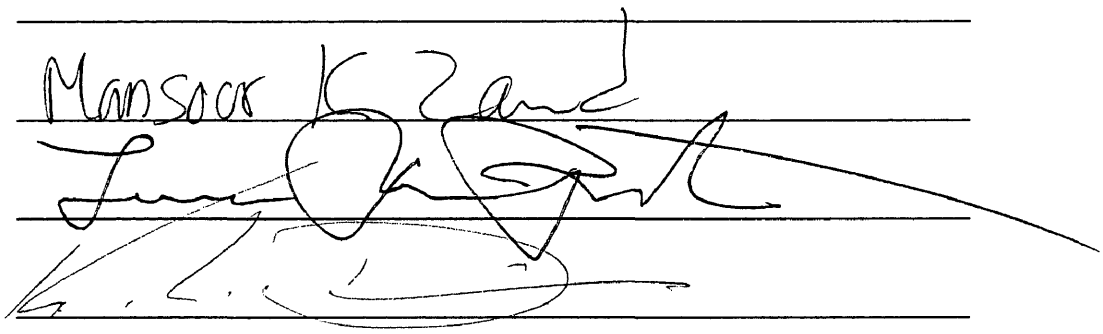
Microform Edition © ProQuest LLC.
All rights reserved. This work is protected against
unauthorized copying under Title 17, United States Code

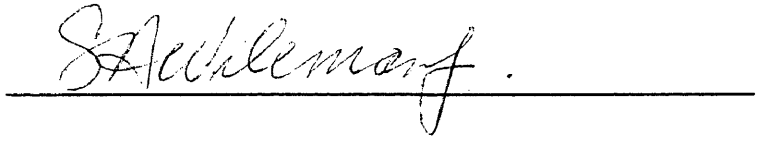
ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

THESIS-EQUIVALENT PROJECT ACCEPTANCE

Acceptance for the faculty of the Graduate College,
University of Nebraska, in partial fulfillment of the
requirement for the degree Master of Computer Science,
University of Nebraska at Omaha.

Committee



Chairperson 

Date April 17, 2003

IMPLEMENTATION E-COMMERCE APPLICATION USING LOTUS DOMINO

Jianrong Dai, MS.

University of Nebraska, 2003

Advisor: Prof. Stanley Wileman

E-commerce technologies enable enterprises to exchange information instantaneously, eliminate paperwork, and advertise their products and services to a global market. The Domino Server family, an integrated messaging and Web application software platform, is easy to build and manage integrated, collaborative solutions. In this project, I build a basic functional Domino-powered e-commerce application named E-Bookstore. The E-Bookstore web site contains three main components of an e-commerce web site (catalog of items, shopping cart, and checkout function), and provides a powerful search function to the customer. The unique session ID for each E-Bookstore web user is generated and stored, and is attached to every item a user adds to his shopping cart. This application also provides the back end maintenance functions such as add book category or book entry. Comparing to popular commercial software, the functions provided in E-Bookstore cover most of useful tools. The E-Bookstore, built on the Domino Application Server R5 platform, has a performance that scales appropriately as the amount of data set increasing and the whole system environment is security.

ACKNOWLEDGMENTS

I would like to thank my advisor Prof. Stanley Wileman for his guidance, support, patience, and understanding during this work.

I would like to thank my supervisory committee members, Prof. Ken Dick and Prof. Mansour Zand and Prof. Leah R Pietron for their helpful guidance.

I appreciate Dr. Simon Sherman for providing me an internship opportunity in his lab.

A special thanks to Mr. Dmitry Shats for his useful discussion on this project.

I would also like to thank the following ITS colleagues at UNO: Steven Lendt, Rachmad Suwondo, and Miun Criffield for their support and friendship

A Special Thanks to my husband Dr. Li Xiao for his love, support and understanding, and my daughter, (Michelle) Xiangting Xiao for her love and bringing me joy.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

1.1 E-Commerce Overview

"E-commerce" is the act of doing business using electronic technology such as intranets, extranets, and the Internet [7]. Commerce is the exchange of money for goods or services between companies or end consumers. Although there are many aspects to commerce and e-commerce, the most common image that the term conjures up is that of a Web-based catalog from which buyers can order products and the sellers can receive payments. Whether you're running a promising start-up or an established enterprise, successful e-commerce means more than enabling customers to purchase products and services online. Global electronic commerce has exploded in recent years. This growth, coupled with rapid changes in information technology and communication, is having a profound impact on business and the workplace. Increasingly, the use of e-commerce is becoming a condition of trade for the manufacturing and retail industries and is imperative for all industries striving to maintain a competitive edge.

The impacts of e-commerce on the workplace are numerous. The Internet provides access to an electronic global marketplace with millions of customers, operating on a 24/7/365 basis. The increasing availability of sophisticated Web tools allows companies to eliminate, re-engineer, and automate business practices, thereby providing a more cost effective, time-efficient manner of conducting business [9]. In spite of prevalent dot.com distrust e-commerce continues to grow. According to a November 2001, Forrester

Research survey of 9,000 online consumers, the high incomes of the most influential Web buyers and their continued online shopping have insulated e-commerce business from much of the economic turmoil [2].

1.2 Domino Overview

The Domino Server family is an integrated messaging and Web application software platform, for growing companies that need to improve customer responsiveness, and streamline their business processes [12]. Domino, the only solution built on an open, unified architecture, is trusted by the world's leading companies to deliver secure communication, collaboration, and business applications [16]. Domino R5.0 servers set a new standard for rich Internet messaging, ease of administration, and integration with back-end systems.

The Domino security model provides user authentication, digital signatures, flexible access control, and encryption. Domino security enables you to extend your intranet applications to customers and business partners. Domino Designer is general-purpose client software featuring an integrated development environment (IDE) that provides easy access to all features of the Domino server.

1.3 Purpose Of This Project

There are so many ways you can build your E-Commerce application, choosing which E-Commerce model is very depends on your budget. If you don't care money, you can buy a set of E-Commerce suit, including web application server, database server, and a high performance database such as Oracle. Since it is a very complex application, you need

pay extra train fee to train your employee, this whole project will cost a lot. But if you do have limited budget, you will concern how can you integrate your applications? How can you share your software and hardware? In our University of Nebraska at Omaha campus, we have Lotus domino R5.0 running as our mail server, we also have some application running on Application server. Can we can build a Domino powered E-commerce application? Yes, we can, and this is my project goal.

1.4 Overview Of The Project

The remainder of this paper is organized into 6 sections. In section 2, I will briefly discuss about E-commerce in general and some terms. In section 3, I will discuss which functions Domino family provides us, why does it suit for power e-commerce application?. In section 4, I will give this application an overview. In section 5 I will describe how to develop a Domino powered e-commerce application and finally section 6 contains the conclusion and future improvement.

CHAPTER 2 E-COMMERCE OVERVIEW

2.1 Introduction

E-commerce is a means of enabling and supporting the exchange of information, goods, and services between companies or their customers [1]. It enables companies to be more efficient in their internal operations and more responsive to the needs and expectations of their customers. E-commerce technologies enable enterprise to exchange information instantaneously, eliminate paperwork, and advertise their products and services to a global market. E-commerce is divided into two categories: business-to-consumer(B2C) and business-to-business commerce(B2B).

2.1.1 Business-to-Consumer (B2C)

The Business-to-Consumer (B2C) e-commerce store model is a publicly accessible Web site offering products for sale. It is analogous to a store on the street, where any member of the public can walk in and make a purchase. A new, unknown customer is called a guest shopper. The guest shopper has the option of making purchases, then providing some general information about them to fulfill the transaction (name, address, credit card, etc.). Most B2C sites encourage users to register and become members. In doing so, the business can establish a relationship with the customer to provide better service, and build customer loyalty.

2.1.2 Business-to-Business (B2B)

The Business-to-Business (B2B) e-commerce store model refers to an e-commerce store specifically designed for organizations to conduct business over the Internet. The two entities are known to each other and all users are registered. B2B applications can streamline operations between businesses. For example, a retailer can place orders from a supplier B2B Web site. This type of e-commerce model greatly increases the speed and efficiency of the buying process between businesses.

2.1.3 Auctions

Auctions can be incorporated into B2C or B2B models. Alternatively, auction e-commerce stores can stand-alone. Sites dedicated to auctions act as brokers, facilitating relationships between buyers and sellers. Auctions are a well-known way of moving surplus merchandise.

2.2 Why use E-Commerce?

Because the internet provides a flexible and dynamic marketplace to exchange goods, services, and information with consumers and business partners, it is becoming increasingly important for business to use the internet to reach new market products both locally and globally. The following list offers some reasons for companies to build commerce-enabled Web sites:

Low entry costs A company can establish itself on the internet, and open for business, with a relatively small investment. Thousands of companies operate simple, inexpensive sites that are successful in their markets.

Reduced transaction costs Dealing with customers over the Web, whether to process orders or to attend to customer support, is cheaper than traditional marketing methods. For example, Dell Computer Corporation estimates that it saves eight dollars each time a customer checks the status of an order at the Dell Web site, instead of calling the company.

Access to the global market With a traditional business, the target market may be local community or, with a higher advertising budget, it may extend to neighboring communities. The Web extends the reach of even the smallest businesses by allowing them to market products globally.

Online distribution The Web enables business to distribute data and software online.

Secure market share A business online protects its current offline market share from being eroded by an online entrepreneur.

2.3 How Does E-commerce Work?

Figure 2.1 shows how the e-commerce process works

The e-commerce process works as follows:

1. A consumer uses a web browser to connect to the home page of a merchant's web site on Internet.
2. The consumer browses the catalog of products featured on the site and selects items to purchase.
3. The selected items are placed in the electronic equivalent of a shopping cart.

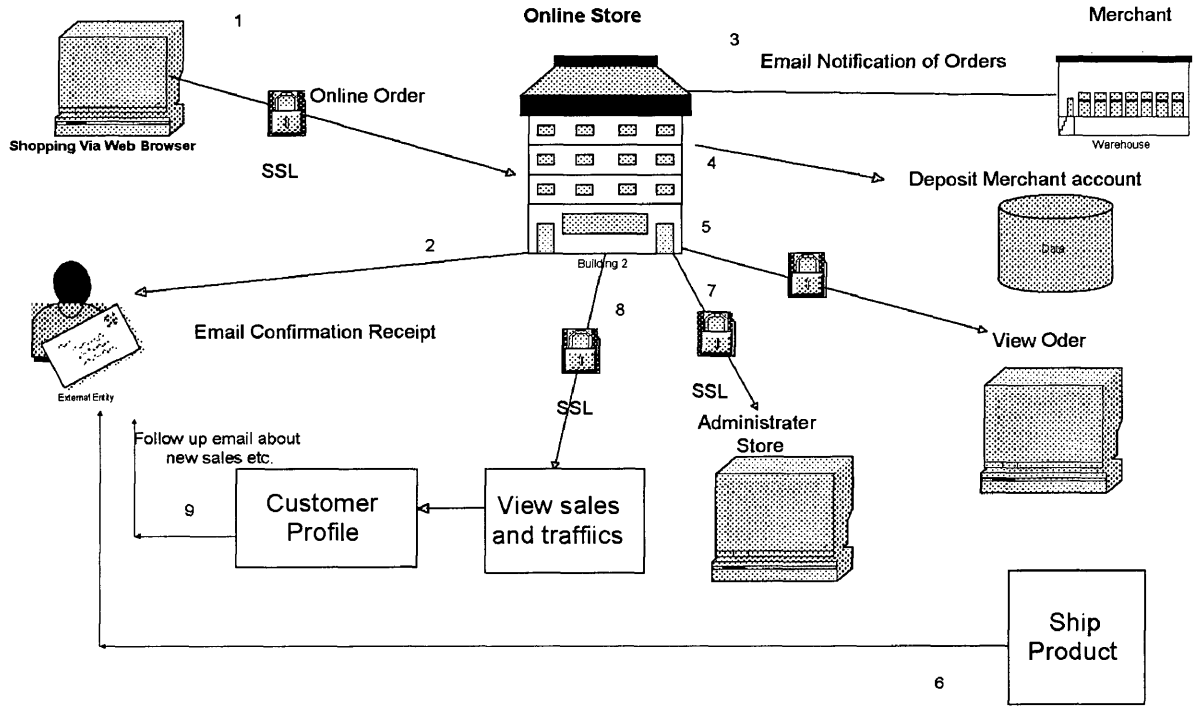


Figure2.1 e-commerce process diagram

4. When the consumer is ready to complete the purchase of selected items, she provides a bill-to and ship-to address for purchase and delivery.
5. When the merchant's Web server receives this information, it computers the total cost of order-including tax, shipping, and handling charges--- and then displays the total to the consumer.
6. The consumer can now provide payment information, such as credit card number, and then submit the order.
7. When the credit card number is validated and the order is completed at the commerce server site, the merchant's site display a receipt confirming the customer's purchase.
8. The commerce server site then forwards the order to a Processing Network for payment processing and fulfillment.

2.4 Some Concepts of E-commerce

Following are some of the key concepts of an e-commerce Web site, it can be help you to understand chapter 4 and chapter 5.

2.4.1 User Profile

Most B2C sites try to maintain information about users. They encourage users to register. Information entered, as well as information gathered during the users' visits from the user profile. The user profile information can be used as a powerful marketing tool, to personalize the Web site content for the user. The personalized content can be used to filter the product catalog for only products that the customer is interested in, or implement such selling techniques as cross-selling and up-selling.

2.4.2 Product catalog

A product catalog on the Web is analogous to a printed catalog. Products are organized into logical groups, and the display of the products is tailored to maximize the sales of the products. Customers can browse the catalog to search for products and then place orders.

2.4.3 Shopping flow

A shopping flow in the e-commerce environment is the process where customers browse the catalog, select products, and purchase the products.

2.4.4 Shopping cart

The metaphor of a shopping cart has become widely used on the Web to represent an on-line order basket. Customers browse an e-commerce site and add products to their

shopping carts. Shoppers proceed to the checkout to purchase the products in their shopping carts.

CHAPTER 3 DOMINO BACKGROUND

3.1 Introduction

Domino is a line of server software that supports your organization's messaging and Web application needs. Domino servers are based on a single architecture, so you can choose the one that meets your current needs knowing it has the flexibility and power to grow when you do. Built on an open, unified architecture, Domino delivers secure communication, collaboration and business applications.

Domino R5 servers set a new standard for rich Internet messaging, ease of administration, integration with backend systems and reliability. Domino helps you leverage your existing investments in people, skills, tools, and backend systems. Now, you don't have to worry about tying together multiple software products for messaging, security, systems management, and data distribution and replication. Domino integrates it all.

3.1.1 History of Lotus Notes and Domino

As you might expect of such complex and successful software, Lotus Notes and Domino share a long and rich history. In some respects, this history mirrors the evolution of the computing industry itself—the development and widespread adoption of PCs, networks, graphical user interfaces, communication and collaboration software, the Web. Notes and

Domino have been there nearly every step of the way, influencing (and being influenced by) all these critical developments.

You may find this a little surprising, but the original concept that eventually led to the Notes client and Domino server actually predates the commercial development of the personal computer by nearly a decade. Notes and Domino find their roots in some of the first computer programs written at the Computer-based Education Research Laboratory (CERL), at the University of Illinois. In 1973, CERL released a product called PLATO Notes. Lotus bought the rights to Notes in 1987. Since then nearly every a couple of years there is a new version was released, from release 1 to release 5, every version have big improvement than prior release to adapt state-of-the-technology features.

3.1.2 Outline of the Chapter

Since the main purpose of the project is to build a domino powered e-commerce application. In this chapter firstly, I will introduce Domino server family and clients, then I will present some advantages of Domino application server, next I will present how does Domino support web application? To better understand Chapter 4 and Chapter5, I will also present Domino database concept and it's components. In the section 6, I will present what kind of programming language Domino supports, and in the last section I will present how Domino solves security issue for web applications.

3.2 Domino Server and Client Concept

Even you are using Lotus and Domino everyday, you may not know all of domino family member. This section presents term of Domino server and client to help you understand why Domino can power E-commerce application.

3.2.1 Domino Server Family

The Domino Server Family allows you to quickly and easily start with what you need today—whether that is messaging or applications—and extend your Domino infrastructure investment whenever you are ready. The Domino server family is comprised of three core servers:

1. **Domino Mail Server**

Domino Mail Server combines full support for the latest Internet mail standards with Domino's industry-leading messaging capabilities— all in one manageable and reliable infrastructure. Its integrated, cross-platform services include Web access, group scheduling, collaborative workspaces, and newsgroups—all accessible from a Web browser or other standards-based client. Domino Mail Server is used for messaging only. Customers who want to deploy their own applications on the Domino server should consider Domino Application Server or Domino Enterprise Server.

2. **Domino Application Server**

Domino Application Server is an open, secure platform optimized to deliver collaborative Web applications that integrate your enterprise systems with rapidly changing business processes. Domino Application Server combines integrated messaging and applications

server. It delivers best-of-breed messaging as well as an open secure Web application platform. The server easily integrates back-end systems with front-end systems business processes.

3. Domino Enterprise Server

Domino Enterprise Server delivers all the functionality of Domino Mail and Application Servers, reinforced with clustering for the high availability and reliability required by mission-critical applications.

3.2.2 Clients for Domino R5.0

Previous versions of Lotus Domino only had one, all-purpose client that would be used by users, administrators, and application developers, with Lotus Domino R5.6, a special client for developers called Lotus Notes Designer for Domino was introduced. As a result of the strong focus on ease-of-use in the design of Lotus Domino R5.0, three individual clients are now available. They are:

1. Notes R5: the user's client

Notes R5 is state-of-the-art e-mail, calendaring, group scheduling, Web access and information management—all integrated in an easy-to-use and customizable environment.

2. Domino Administrator R5: the administrator's client

Domino Administrator is a new, integrated administration control panel that provides simple, yet flexible administration for your Domino environment.

3. Domino Designer R5: the developer's client

Domino Designer R5 is an integrated development environment. It enables developers to rapidly build secure Web applications that incorporate enterprise data and streamline business processes. Most of the functionality in Lotus Domino can also be accessed from Web browsers. In the following chapter, I will concentrate the description on the Domino Application Server used for project tests.

3.3 What's Advantage of the Domino Application Server

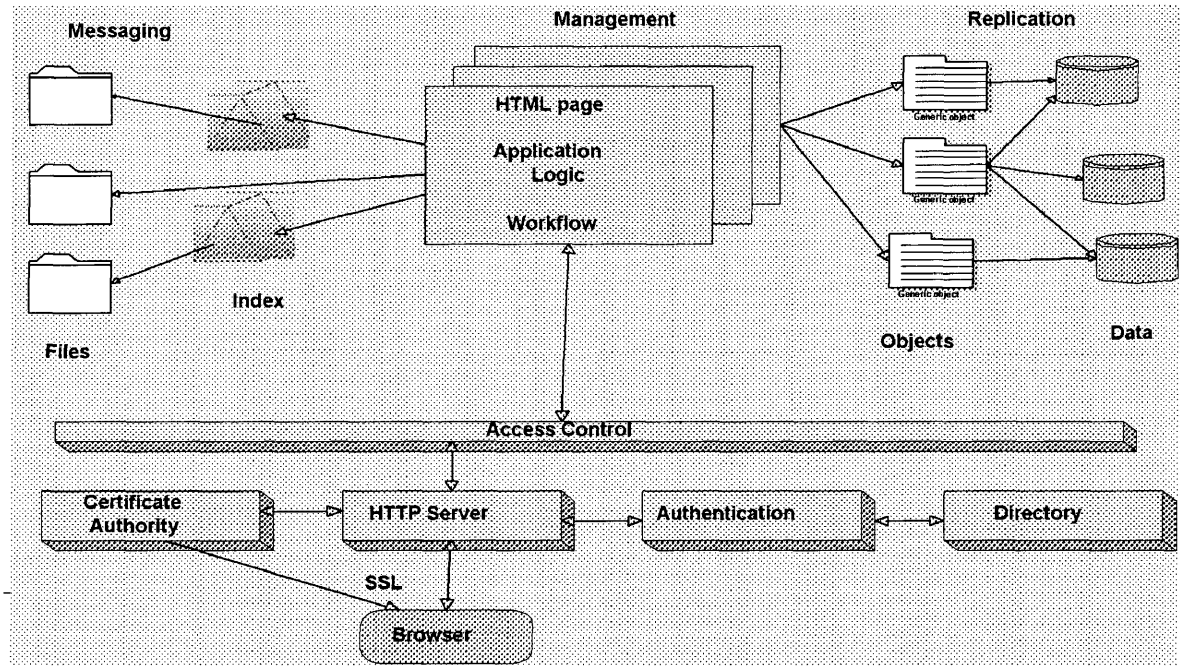
As I mention before, I intend deploy this application on to application server, this section I will discuss Domino application server environment.

Domino Application Server R5 allows you to integrate your Domino applications with the enterprise systems. It leverages current information assets with built-in connection services for live access to relational databases, transaction systems and ERP applications. It is optimized for collaboration and provides comprehensive application services like workflow and messaging, so you can easily build and manage integrated, collaborative solutions.

You can deploy and maintain the applications with its integrated development tools, standards support and unmatched server-to-server replication. But Domino is also open as you can use your favorite HTML authoring tools, Java IDEs and scripting tools to create Domino applications.

Domino Application Server is an open, secure platform optimized to support rapid delivery of collaborative Web applications that integrate your enterprise systems with dynamic business processes. Domino Enterprise Connection Services (DECS) provides

rapid connectivity to enterprise data using a visual mapping interface. Figure 3.1 shows the different services of a Domino application server available to Web applications [12].



CORBA/IIOP support lets you integrate Domino with your application's architecture. This support extends Domino application services to Web clients allowing you to serve Lotus Notes clients and Web browsers with the same application.

With its comprehensive development environment, the Domino Application Server lets you move beyond static Web sites—to create high-value business solutions that include workflow, content management and highly flexible security. With Domino, you can easily create self-service applications like e-commerce and customer care, and connect them to back-end systems.

The flexible security of Domino allows you to personalize access to data and applications based on individual and group roles. It is also extended to HTML files and other data, for pervasive security no matter how or where Web content is stored.

The Domino R5 HTTP engine delivers outstanding performance and Java servlet support.

3.3.2 Services Offered by Domino Application Server

Domino Application Server offers the following services:

1. **Object store**

Documents in a Domino database can contain any number of objects and data types, including text, rich text, numerical data, structured data, images, graphics, sound, video, file attachments, embedded objects, and Java and ActiveX applets. The object store also lets your Domino applications dynamically present information based on variables such as user identity, user preferences, user input, and time.

2. **Search engine**

A built-in full text search engine makes it easy to index and search documents stored in Domino and files in the file system.

3. **Security**

Integrated X.509 support lets you register new users with Notes and/or X.509 certificates. S/MIME support ensures message integrity for all client types; SSL V3 for IIOP and LDAP clients. Authentication via trusted third-party directories reduces complexity and duplication of information.

4. **Directory**

The directory supports a multi-enterprise infrastructure of any size and integrates with other directories via full support for LDAP V3, the open standard for directory access. Its extensible schema allows you to store any information you choose.

5. Workflow

With Domino workflow support, you can define processes to route and track documents, to coordinate activities both within and beyond your organization.

6. Messaging

An advanced client/server messaging system with built-in calendaring and scheduling enables individuals and groups to send and share information easily. Message transfer agents (MTAs) seamlessly extend the system to Simple Mail Transfer Protocol (SMTP)/Multipurpose Internet Mail Extension (MIME), X.400, and cc:Mail messaging environments. The Domino messaging service provides a single server supporting a variety of mail clients: Post Office Protocol V3 (POP3), Internet Message Access Protocol V4 (IMAP4), Message Application Programming Interface (MAPI), and Lotus Notes clients.

7. Development environment

Domino Designer is optimized to work with Domino, and features a complete set of visual tools for rapid development and deployment of secure, e-business solutions. It supports your favorite tools for HTML authoring, Java development, and scripting.

8. Domino objects

Domino offers a collection of software objects that expose Domino functionality to several language bindings including Java, JavaScript LotusScript, and due in 1999, OLE

and COM. This allows you to switch programming languages without having to learn new ways to program for Domino.

9. Domino Enterprise Connection Services (DECS)

Domino Application Server includes DECS, for live access to enterprise systems. DECS supports a wide range of enterprise systems, including DB2, Oracle, Sybase, ODBC, EDA/SQL, SAP, PeopleSoft, JD Edwards, MQSeries, CICS, and more. Without programming, DECS allows you to create Web applications that access or update enterprise data in real-time, via persistent, parallel, pooled connections.

3.3.3 Domino Application Server delivers reliability and manageability

Domino Application Server delivers reliability and manageability with:

- Transactional logging for Domino databases
- Backup support and APIs to allow tight integration with third-party backup tools on all Domino platforms, including NT, UNIX, AS/400 and S/390
- High availability services such as online indexing and database compaction, fast server restart and more
- Remote server management options
- Centralized control of Notes desktops
- Mail server capabilities

3.4 Domino and Web Support

Since this project is an E-commerce application, this section I will talk about how well Domino support web application.

As shown in Figure 3.2, Domino adds to the Lotus Notes server an HTTP task that allows Domino to serve both Notes clients and Web browsers. Domino automatically converts all Notes design elements and documents to Web HTML pages, allowing users access to Notes databases either through a Notes client or a Web browser.

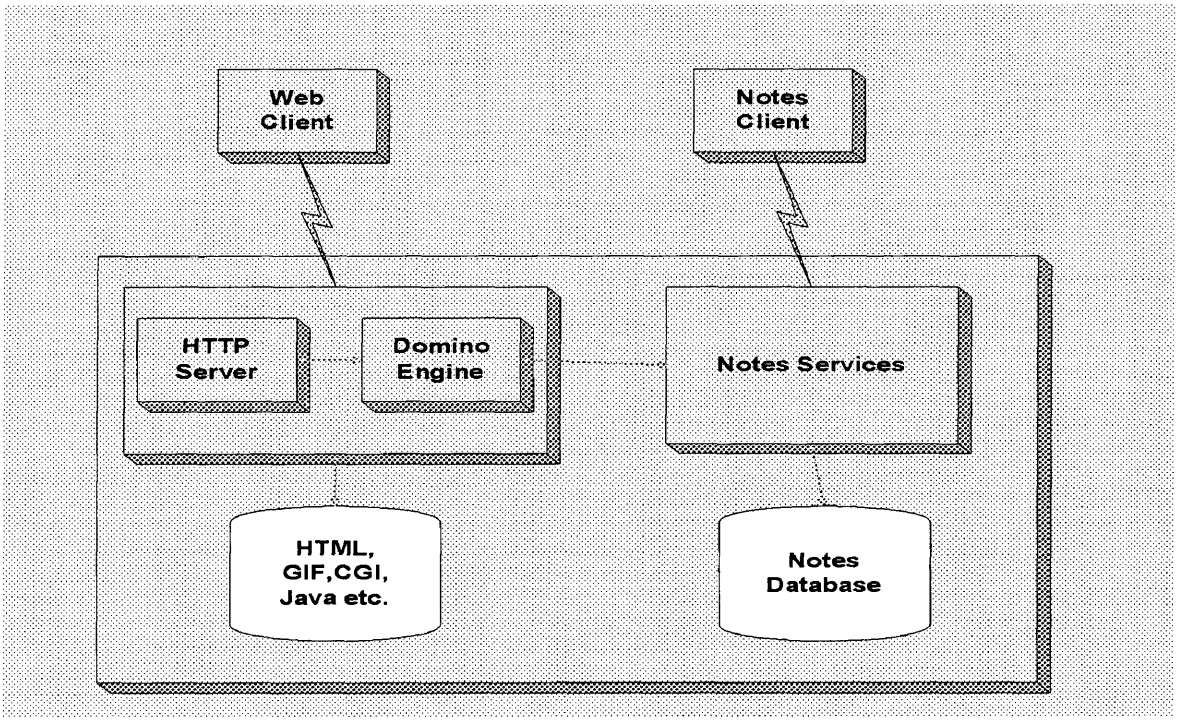


Figure3.3 Domino Architecture

The Domino Web server examines the URL in the incoming request and determines if the request is for an item in a Domino database or if it is a request for an HTML file in the file system.

- If the request is for an HTML file, Domino acts just like any other Web server and serves the file to the Web client.

- When the request is for something in a Domino database, Domino interacts with the database to serve the information to the Web client or to put information from the Web client into the database.

Domino supports URL extensions that expose functionality to the Web client. Domino URL commands have the syntax:

```
http://your.web.server/DominoObject?Action&Arguments
```

where:

- your.web.server is a Domain Name Server (DNS) entry or an IP address.
- DominoObject is a Domino construct (a database, view, document, form, navigator, agent, or other).
- URL commands for accessing Domino objects use the following syntax: `http://your.web.server/Database/DominoObject?Action&Arguments`
- where: Database is the database in which the Domino object resides.
- Action is the desired operation on the specified Domino object, for example: ?OpenDatabase, ?OpenView, ?OpenDocument, ?EditDocument, ?OpenForm, and other.
- Arguments is a qualifier of the action, for example, Count = 10 combined with the ?OpenView action limits the number of rows displayed in a view to 10.

3.4.2 Scalability

One of the reasons Domino makes an excellent choice for building web application is its scalability. Domino has a long history of being supported on many operating systems. In

the current version, these include Windows NT [20], Windows 2000 [20], Windows XP[20], Linux[20], Unix (HP,AIX) [20], and OS400[20].

For companies running multiple Domino servers, the servers don't have to be on the same platform; it is common to see a mixed environment. Since IBM purchase Lotus, IBM has positioned Lotus as the lower-volume web server choice and WebSphere as the higher-volume choice. However, Domino running on a high-end Unix, Linux or OS400 system is quite capable of handling an immense volume of traffic. The additional ability to cluster servers together, even across different operating systems, allows Domino to scale even further.

There are many compelling reasons to consider Domino for your business' web applications. I haven't mentioned cost – a Domino server license can be purchased under $2000, with a reasonable investment, you can be well on your way to leveraging the power, the flexibility, and scalability of Domino in your business.

3.5 Domino Database Components

With Domino, the Web site is organized through Notes databases designed in the Notes Object store format. A Web database is something different from a Notes database if only for the viewing mechanism and for the fact that it resides on a server running the HTTP server task.

Domino databases manage documents, as opposed to relational databases, which manage tables. Unstructured or structured data as well as associated programming logic are stored within the database. Documents can contain any number of object data types including

text, rich text, numerical text, structured data, images, graphics, sound, video, file attachments, embedded objects, Java and activeX applets.

3.5.1 What's in Domino Database?

A database contains three basic components: forms, fields and views. In addition, icons, help documents, navigators, agents, sections, actions, formulas and scripts play an important role in giving an application-sophisticated automation and processing power. In Domino R5, Lotus added some new features such as outline, pages, and frameset. This section will explain each component in detail.

3.5.1.1 Documents

A Domino document is a container for data. Each document is effectively a collection of completed fields. A unique identifier is given to each document upon its creation, and is used by the Domino database as a primary key.

A document's design is held within a Domino form, which specifies what a document can contain in each field, for example, a number, a date, text, or rich text.

In Web applications, documents are individual Web pages that store the data. Documents can contain text, graphics, fields, applets, HTML, embedded navigators and embedded views.

3.5.1.2 Fields

A field in a document contains one piece of information. In Web applications, fields can also contain Web pages, multimedia objects, graphics and files.

3.5.1.3 Form

Forms are design elements that give users a framework for entering new information in a database and for viewing existing information. It is possible also to associate forms with other design elements, such as views and navigators, to create special Web effects.

3.5.1.4 Navigators

Navigators provide a graphical way for users to find documents or take actions in an open database without opening views. Domino converts navigators to HTML image maps on the Web. A navigator can be an entry point for a database if you design it to open automatically when a database opens.

3.5.1.5 Views

Views are used to sort, list, filter and categorize documents. Application developers design views with particular selection criteria so that they will display lists of pertinent documents in a given order. Views allow users to navigate multiple documents. Documents are accessed and opened from within views. Views also contain other design elements, such as columns, buttons, and programming code, which allow users to manipulate or change multiple documents.

3.5.1.6 Templates

A template is a pre-designed database that you can use to create new databases quickly. When creating a database from a template, you receive a number of forms, views, and navigators that determine how the database looks and functions.

3.5.1.7 Agents

Agents allow you to automate many tasks within Notes. For example, they can be used for changing field values, sending mail messages, deleting documents or interacting with external applications. Users in the foreground or automatically in the background can run them. Agents can call other agents, run on servers, or on workstations. Therefore, the user can use them to easily access, process, and manage data on other servers or in other databases. They can be easily distributed because they can be replicated.

3.5.1.8 Outline

An outline is the skeleton of your application, where each entry represents a key piece of the application. When the outline is embedded on a page or form, users can click on the outline entries to take them where you want them to go. Outline entries are fully programmable. You can add logic that controls how entries are rendered by the Notes client or Web browser.

By combining the outline with framesets, you can easily create a powerful application interface for navigating your site. Your application can include multiple outlines, which launch links in target frames.

3.5.1.9 Pages

A page is a database design element that structures and displays information, including text, graphics, applets, and links. A page might include HTML that you copy and paste or import, or it might be entirely composed using the tools available through Designer, such as the text editor that lets you enter and style text without entering HTML. For example, a

home page for a Web site typically displays information about the site and provides a navigational structure for accessing other parts of the site. A page can contain text, tables, graphics, image maps, applets, links, embedded controls, such as an outline control, navigators, views, or folders, OLE objects and custom controls.

Pages are part of an application's design, and can be inherited from a template. Pages can be included in a full-text index, but they do not display in views. You must explicitly display a page using a link or a programmed action.

3.5.1.10 Frameset

A frameset is a collection of multiple frames. A frame is one section, or pane, of the larger frameset window and is independently scrollable. By using framesets, designers can create links and relationships between frames. Framesets provide the ability to leave one page displayed as users scroll or link to other pages or databases.

3.6 Programming for Domino

Domino provides a variety of development alternatives for creating applications, thus enabling you to choose the most appropriate product. The Lotus Notes Designer client is the native development environment for Domino using LotusScript and LotusScript extensions (LSXs) as the development language. However, Lotus has also developed a set of APIs for C, C++, and Visual Basic. Third-party vendors (PowerBuilder, Gupta SQLWindows) have worked together with Lotus to integrate their own development tools.

3.6.1.1 Formulas

You can use the Lotus Notes formula language to develop applications. The formulas are attached to certain objects in Domino, such as selection formulas for views. The formula language contains a set of built-in macros, functions, and commands. The functions, referenced as @Functions in Lotus documentation, are prepackaged formulas. Domino has more than 100 functions.

The @Commands are built-in functions that enable you to programmatically simulate menu sequences.

3.6.1.2 LotusScript

LotusScript is an embedded, BASIC scripting language with a powerful set of language extensions that enable object-oriented application development within and across Lotus products. Its interface to Notes is through predefined object classes. Notes oversees the compilation and loading of user scripts and automatically includes the Notes class definition. Two types of Notes object classes are provided:

- Front-end user interface classes UI classes provide features to emulate user actions. They also provide access to objects such as workspace, database window, field and rich-text field.
- Back-end classes Back-end classes represent Notes objects such as database, view, agent, document, and item. You can use these classes to manipulate Notes elements directly in LotusScript.

3.6.1.3 Java

Domino is a complete Web application server, which fully supports the Java environment. Domino applications can be written in Java as you can call Domino objects from a Java program. Domino supports JDBC calls to allow Java programs access to Domino data. Domino R5 also supports CORBA to build distributed Domino applications.

3.6.1.4 Other Languages

Domino R5 supports many Internet-models of programming, so you can choose your favorite language when designing Web applications—whether that language is JavaScript, Java, HTML 4.0, or LotusScript

With native support for JavaScript and HTML in the Notes client, you can now design applications that run the same on the Web as they do within Notes. In addition, Domino R5 allows you to use third-party design tools, such as NetObjects Fusion, NetObjects ScriptBuilder, and IBM VisualAge for Java.

3.7 Domino SSL implementation

Implementing e-business often means that you will, at some level, exchange sensitive data between the clients and the servers. This is when security becomes an issue. How well funded Domino or Notes is when it comes to security will be covered in this section, where I reveal:

- The strong security infrastructure that has always been part of Notes and Domino
- How Notes and Domino R5.0 support today's open Internet security standard (X.509).

These elements are the basics in the Domino security infrastructure and cover the client security, the server security, and the transmission security when sending data between the clients and the servers.

There are two types of clients that I cover: the traditional Lotus Notes client and a Web browser client. They are two totally different types of clients, but are authenticated by the same security system that has been implemented in Notes and Domino. The exception is the validation and the use of certificates when securing the transmission.

Lotus Notes clients and Domino has their own way of dealing with certificates compared to the way a SSL connection uses certificates when communicating between a Web server and a Web browser. It is important that you understand how the Domino security works so you can implement the right level of security. I will go through the basics of Domino security, followed by the basics of using secure connections between clients and servers.

When talking about security in Notes and Domino, you can look at it as nine layers. The upper layer is simply getting access to the network where the Domino server is placed. The lowest layer involves security at the field level within a Domino document. The nine layers, from top to bottom, are:

- Network
- Domino server
- User authentication
- Database
    - Views/forms

- Document
- Section editor
- Hidden paragraphs
- Edit fields

3.8 Conclusion

Domino Built on an open, unified architecture and integrates message, security, systems, management, and data distribution and replication together. All these services make sure we can build a total Domino powered E-commerce application.

CHAPTER 4 APPLICATION DESCRIPTION

4.1 E-Bookstore Overview

E-Bookstore is a demo bookstore to sell book online. The website can be access all over the world. In this section I will present an overview of function of the E-Bookstore application. First I will present web function and client function, then I will present security consideration.

4.1.1 The Customer View

The E-Bookstore application provides an interface for customer to do following:

- Browser E-Bookstore book catalog.
- Assign a unique Cart ID for each customer.
- Search Book entry.
- Create a new purchase order.
- Submit a new purchase order for processing.
- View a report on the status of POs they have submitted.

This interface can be accessed online only through a suitable Web browser.

4.1.2 The Staff View

Within E-Bookstore, staffs will be assigned to do following:

- Maintain book catalog for example add new book entry, delete old book entry.
- They use their Notes client to view the purchase orders.

- View the customer credit summary.
- Evaluate new purchase orders. Staff can either reject the order and an e-mail is sent to the buyer explaining the reason for rejection, or accept it and the application generates the workflow to process the order such as inventory pick lists, packing slips, shipping labels, and invoices.
- Notify customer that item is available now.

4.2 Domino Client Security

Domino security applies to all clients, but works somewhat differently for each. In general, there are a number of levels of security, starting with server access, then database access, View and Form access, and continuing through a number of levels down to field level access. In this application I am mainly concerned with Form and Document access control. These are controlled through a combination of three elements:

- Person (and other user types) documents in the Domino directory and signed certifier files containing user passwords
- ACL for the database. This list includes a default entry and can be populated from the Domino directory. Each entry can be assigned one of the following security levels, listed in order of access level:
    1. Manager (sets database security and other settings)
    2. Designer (creates views, forms, agents and other elements)
    3. Editor (edits most documents)
    4. Author (creates documents, edits own documents)

5. Reader (reads documents)
6. Depositor (creates documents but cannot read any)
7. No Access

Managers can also create security objects called *roles* in the ACL. Once a role is created, it can be assigned to one or more entries in the ACL. This is useful, for example, in controlling form and document security.

- Security settings for the form. These settings allow you to control which entries from the ACL, including roles, can create documents with the form and which entries can read those documents.

The database has a default access level of Author, but access to the Order form and documents is restricted. Any user can create *Order* documents, and users with the Admin role can read *Order* documents. Only users with the OrderAdmin role are able to edit Order documents created by customers.

Notes Client users need to have an ID file in order to use the Notes Client. Generally, they are authenticated when they try to access the server. Domino can tell who the user is by the ID file they are using, so they are only required to enter a password. The actual authentication process is beyond the scope of this project but, when implemented correctly, is very secure.

In the case of Web client users, since I am creating an international web, I permit Anonymous access this application, I don't want to let Domino authenticates them when they try to access this application. I assign each customer a unique ID when they first

visit this application to trace the whole session, I will discuss this technique in detail on next chapter.

4.3 Performance Considerations

When talking about performance, people often have different perceptions of what it is. Here are two ways to look at performance:

- Time aspect — Responsiveness

  Performance is a measure of time spent on an individual operation or job.

- Capacity aspect — Throughput

  Capacity or throughput can be another measure of performance. Performance can be measured in terms of data transfer rates, generally expressed as transactions per hour. It can also be measured in terms of resource utilizations such as CPU utilization or disk storage utilization. It does not make sense to talk about performance without mentioning throughput capacity or scalability. An application may perform very well during a pilot deployment, where only a subset of the targeted user population works with it — and then fail badly when rolled out to all users.

4.3.1 What Determines Performance in Domino Application?

The responsiveness and throughput you can reach in Domino application is a product of the following [22]:

1. Application design and implementation,
2. Potential interaction with external systems,

3. Domino Server configuration,
4. Network configuration,
5. Server and client operating system configurations,
6. Server and client hardware configurations.

Often the main factors in a long response time or unacceptable throughput is found in some of the layers below the application, and how to configure Domino Server and the network are not in the scope my project. However, when I designed this project, I applied some technology to improve the performance of my application. For example I split up the application in two databases to support more users and get better response time. I chose suitable programming languages to code the different functions of the application to obtain suitable performance.

CHAPTER 5 APPLICATION DEVELOPMENT

5.1 Introduction

The development of application components will be divided into the following five sections:

1. Initial Domino Development.
2. Site navigation -- a challenge faced by all e-commerce application. How do you get your customers to the catalog items that interest them?
3. Session tracking – E-commerce solutions of all flavors have one thing in common: through a variety of mechanisms, each customer is assigned a unique identifier so that the action of that individual can be recognized throughout their interaction with application.
4. Add-to -cart capabilities.
5. Check out scheme – tackle the shopping cart and whole ordering process, including the finalization of cart contents, collecting customer information, and basic credit card processing.

5.2 Initial Domino Development

Domino design components cannot exist outside of a Domino database, so before I can create components I need to create a Domino database to put them into.

5.2.1 Database Creation

In this application I created two Domino databases called ebook.nsf and oders.nsf, ebook.nsf stores book catalog and front-store design components, and oders.nsf stores customers order information such as customer name, credit card, ordered book name and quantities etc.

5.2.2 Two kind of users

There two kinds of user in this application the client user and the web user. The client users are users access application using Lotus Notes client software. Client users are staff to maintain website database for example add new book entry, delete old book entry, process credit charge etc. the web users are customers, they can access application wherever using IE browser or Netscape browser.

5.2.3 Implement Security

By default, Web users are identified as Anonymous when they access the Domino server and all the read components of the database. To allow anonymous access to the catalog but to force identification to the other part of the application, set up some roles for the users. To set up roles, do the following:

In the ebook database, select **File - Database - Access Control**

1 Select the **Roles** icon of the Access Control List dialog.

2 Click **Add** and enter the name *Admin* and click **OK**. This is the role for book store staff.

3 Select the **Basics** icon of the Access Control List dialog.

4 Select staff name on the People, Groups, Servers list. In the Roles list on the bottom right corner of the dialog, select *[Admin]*.

5 Click **OK** to close the Access Control List dialog

6 In the orders database, select **File - Database - Access Control**

7 Select the **Roles** icon of the Access Control List dialog.

8 Click **Add** and enter the name *Admin* and click **OK**. This is the role for book store staff.

9 Click **Add** again, enter the name *OderAdmin* and click **OK**. This is the role for credit evaluator.

10 Select the **Basics** icon of the Access Control List dialog.

11 Select staff name on the People, Groups, Servers list. In the Roles list on the bottom right corner of the dialog, select *[Admin]* and *[OderAdmin]*.

12 Click **OK** to close the Access Control List dialog.

5.3 Client User Interface

This application contains two Lotus Notes database, one is used to store book information, the other one is used to store credit information. To access this application, client users need install Lotus Notes client software, and for safety, only have admin role users can access this application. So before access application from client, make sure your name is added to database Access Control List, and assigned admin role.

Click **File** ->**Database** -> **Open**, select database name called e-bookstore on the local, click **Ok**, you will see Figure 5.1 screen. Some important views locate in the left screen,

and each view contents are shown in the low right screen. On the up right column, there are application title and bookstore log. Actually, client interface function as back-end system, from this interface you can add new catalog of book, add new book entry, delete old book entry, process credit charging, and handle book shipping.

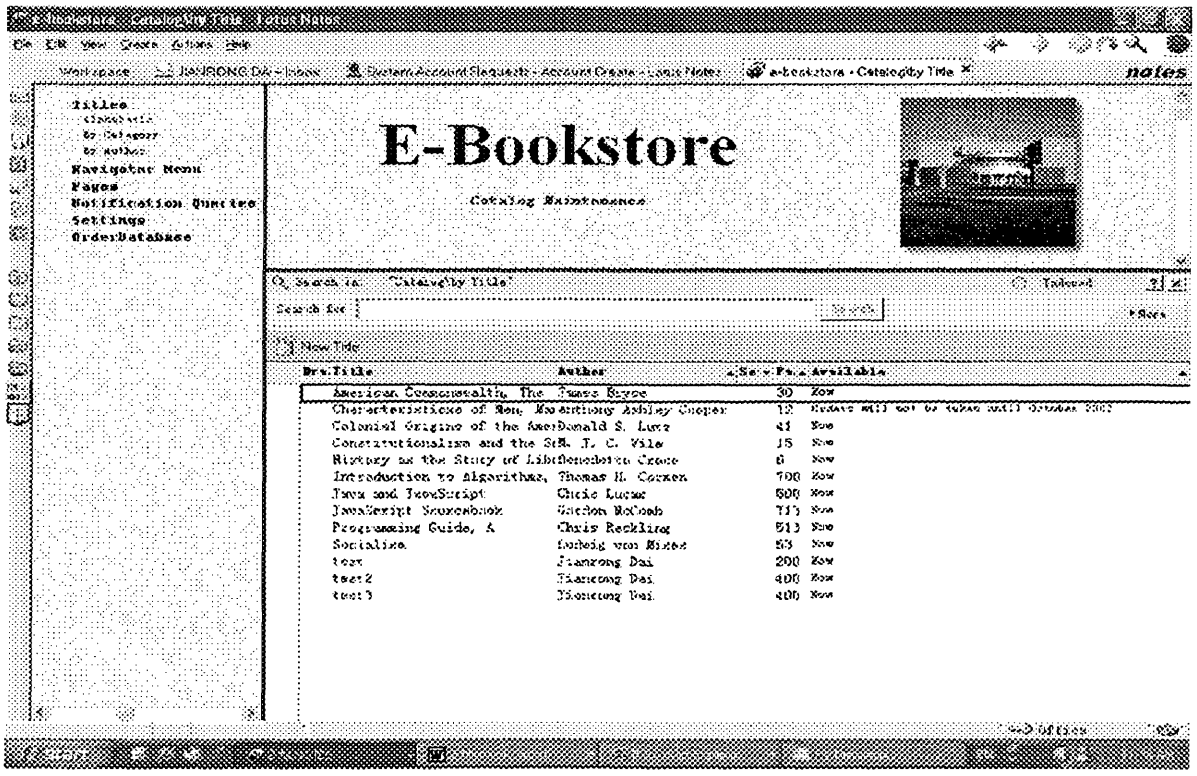


Figure5.4 E-bookstore client interface

5.3.2 Steps to Add New Category

1. Double click E-bookstore icon from workspace, on the left navigator, click "Navigator Menu", you will see Navigator Entry view on the right frame. See Figure 5.2.

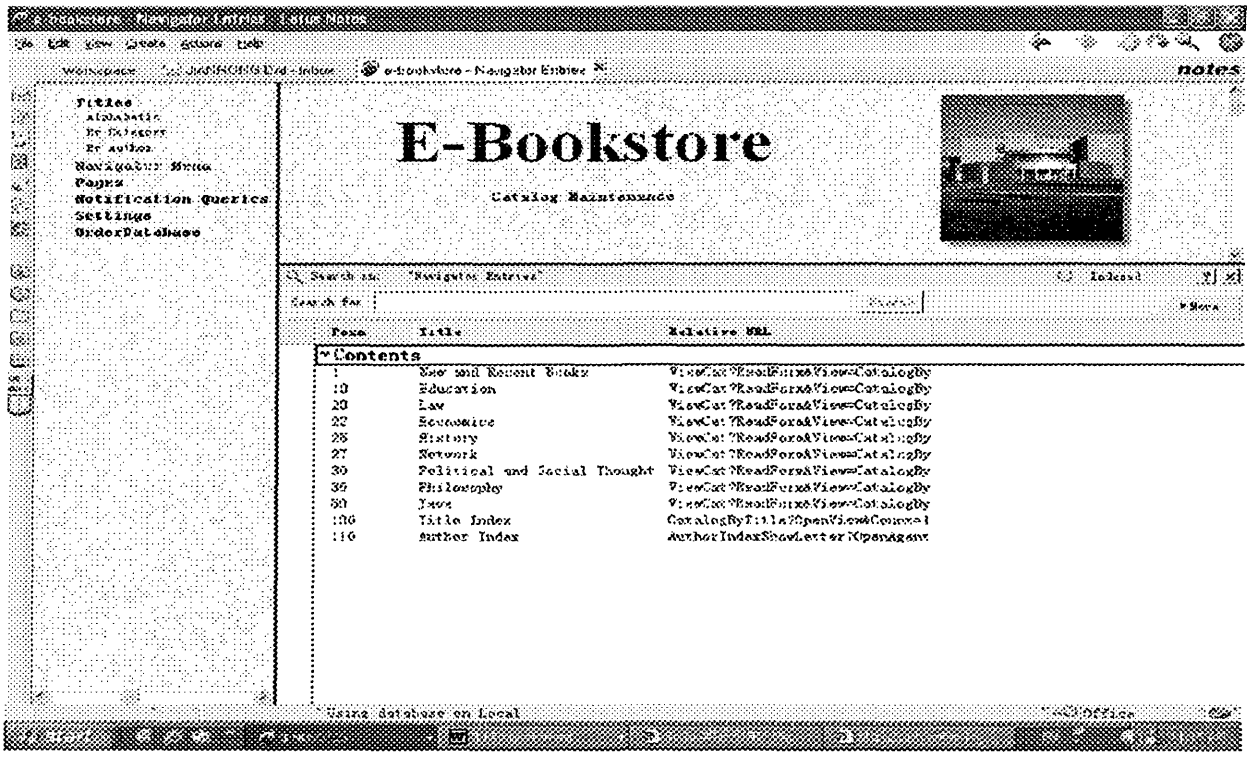


Figure5.5 Navigator Entry view

2. Click "Create New Navigator Entry" button located in the menu bar, you will open a new form, let you fill new navigator content. For example, if you want add a new navigator category "Network", you can fill form like Figure 5.3, then click Save/Close button, thus your new category will show up on the web. If you want to test, click "Pages" on the left navigator, switch to the Pages view. Open the first document, E-bookstore home page. Choose Actions - Preview in Web Browser. You will see screen similar to Figure 5.4, you can see that new catalog " Network" does show up on the web.

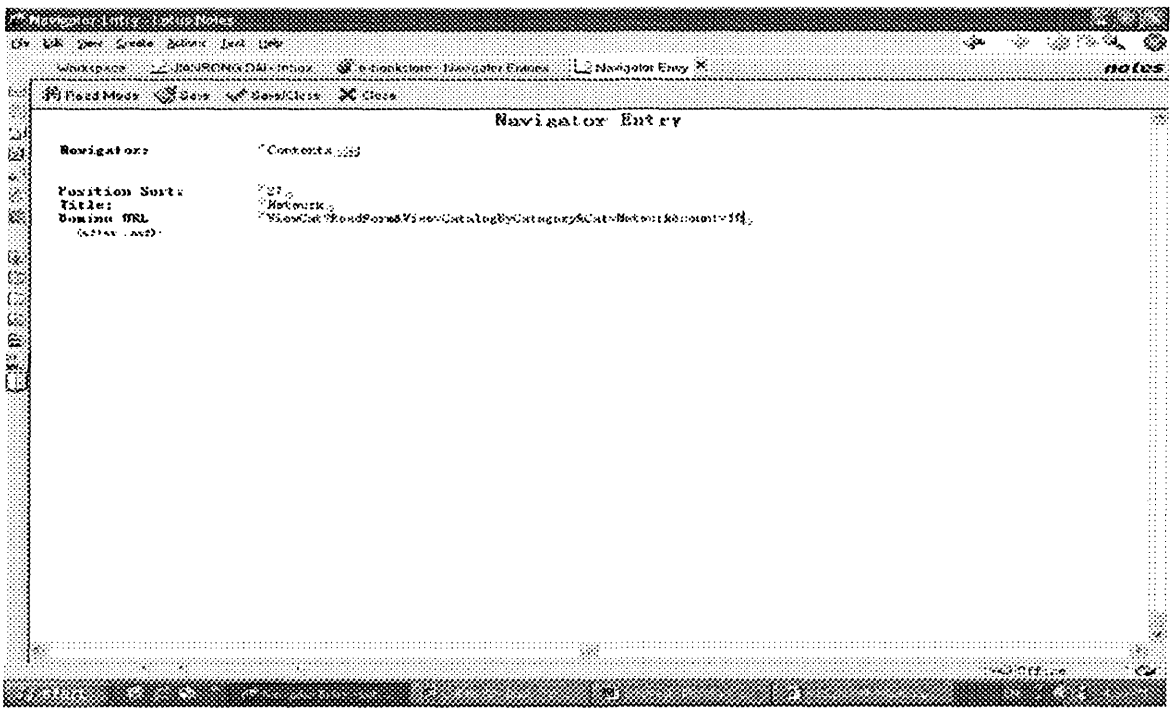


Figure5.6 Create New Navigator Entry

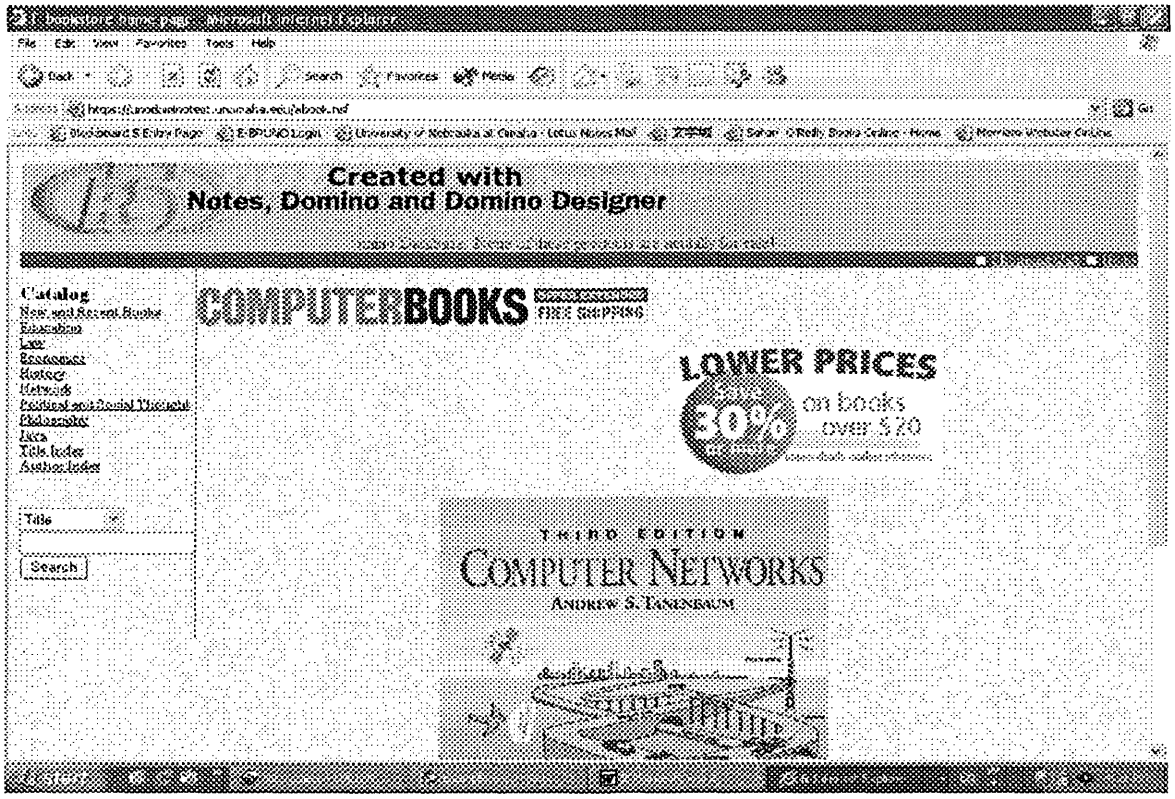


Figure5.7 E-bookstore home page

5.3.3 Add New Category Technical Design

To implement above function, a form called "Navigator Entry" was created. Table 5.1 list fields and text on this form.

Table 5.1 Navigator Entry Form Fields and Text

Static Text (column 1)	Field Name (column 2)	Field Type
Navigator	Navigator	Dialog list, editable
Position Sort	PositionSort	Nummer,editable
Title	Title	Text, editable
Domino URL	RelativeURL	Text, editable

This form contains four fields: Navigator, a text field that allows me to categorize the links; NavigatorPosition, a numerical field used to sort the documents in their respective categories; Title, which is text that serves as the link text seen by customers; and lastly, RelativeURL, another text field that will hold the URL of individual link destinations.

Once client user add a new catalog using Navigator Entry form, new catalog should be shown on the web immediately. The goal is the allowance for look-and-feel and content changes with as little modification to the actual application code as possible. With this in mind, I created a hidden view called (NavigatorEntriesHTML, see Figure 5.5:

The first column is categorized on the Navigator field. In this application, this field contains Contents across the documents. The second column sorts on the NavigatorPosition field and will control the order in which the link documents appear on

the web. The third column represents the @DbLookup's target: the HTML representing the individual links. A single document's value for this column looks like this:

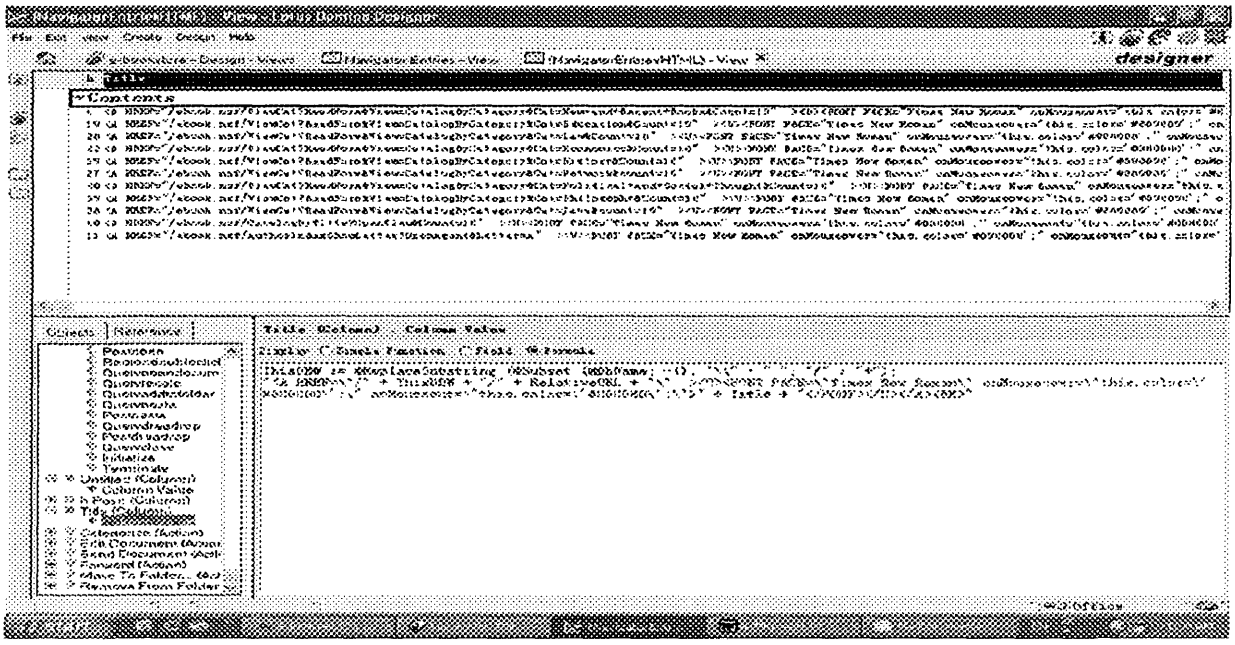


Figure5.8 NavigatorEntriesHTML view designer

<A HREF="/ebook.nsf/ViewCat?ReadForm&View=CatalogByCategory&Cat=Network&Count=10"><U><FONT FACE="Times New Roman" onMouseover="this.color='#800000';">Network</FONT></U></A>

The formula for the third column is:

```
ThisDBW:=@ReplaceSubstring(@Subset(@DbName; -1); "\\" : " "; "/" :"+");
"<A HREF=\"/"+ ThisDBW + "/" + RelativeURL + "\" ><U><FONT FACE=\"Times New Roman\" onMouseover=\"this.color=\'#800000\';\" onMouseout=\"this.color=\'#000080\';\">" + Title + "</FONT></U></A><BR>"
```

The first line is simply setting a temporary variable equal to the path of current database.

The path of current database is used in HREF of link tag - <A></A> - on the second line. To complete the HREF, the formula then refers to the RelativeURL field on thelink documents, which in the case of the HTML link above look like:

```
ViewCat?ReadForm&View=CatalogByCategory&Cat=Network&count=10
```

Next, the formula specifies the link's font and its onMouseover and onMouseout properties used to control the link's color as the mouse passes over it. Finally, the link document's Title field represents the actual text displayed to the customer by the link, in the link above, it's Network.

Putting all these steps together results in set of reusable, dynamic navigation links and will bring a great deal of flexibility to web site.

5.3.4 Add New Book Entry

Whenever a book ready online to sale, staff need to create a book entry document. Steps as following:

1. Open the E-bookstore database (ebook.nsf), from Notes client.
2. Choose Create – Catalog Entry.
3. You will open Catalog Entry Form, fill each field one by one shown on Figure5.6.
4. Click Save/close button.

5.3.5 Add New Book Entry Technical Design

I design two forms both called "Catalog Entry", one for notes client users, the other for web users. When you develop Web enable Domino application, separating form design is most common method. In this application, client user use client form to create, edit book

entry information. On the other hand, web users can only use web form extract information from client form, and web users can't edit book information, they can only read. Catalog Entry form design snapshot see Figure 5.7, its field and text see table 5.2.

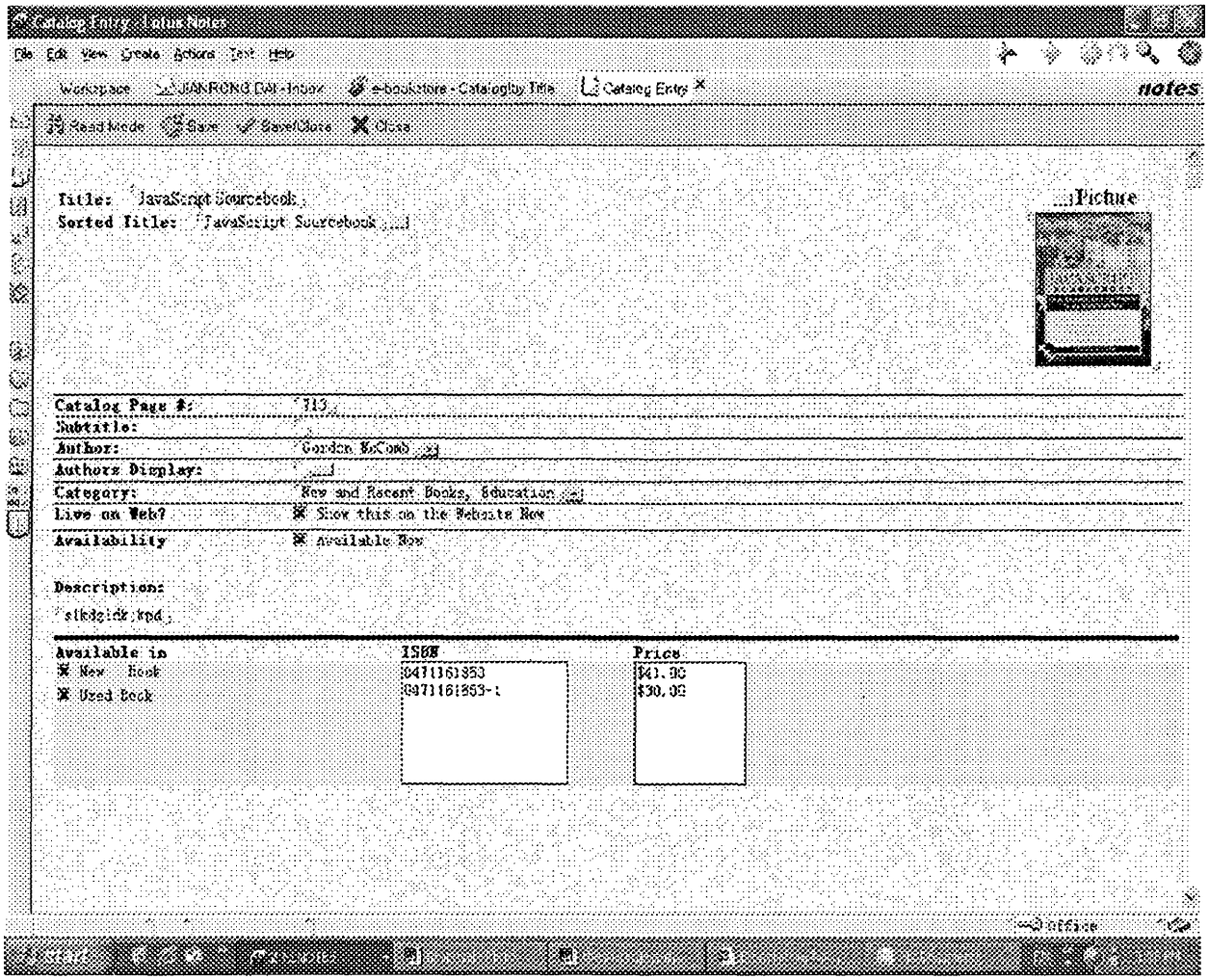


Figure5.9 Catalog Entry Form interface for client user

Table 5.2 Catalog Entry Form for client Field and text

Static Text (column 1)	Field Name (column 2)	Field Type
Title	Title	Text, editable
Sorted Title	TitleSorted	Text, editable
Picture	Picture	Rich Text, editable
Subtitle	Subtitle	Text, editable
Author	Author	Text, editable
	AuthorLNF	Text, computed
Authors Display	AuthorsDisplay	Formula, FIELD AuthorsDisplay := "by " + @Implode (Author; " and ");
Category	Categories	Dialog List, editable
Live on Web?	Live	Checkbox, Editable
Availability	Available	Checkbox, Editable
Description	Body	Rich Text, editable
Available in	MediaTypes	Checkbox, Editable
ISBN	MediaISBNs	Text, editable
Price	MediaPrices	number, editable

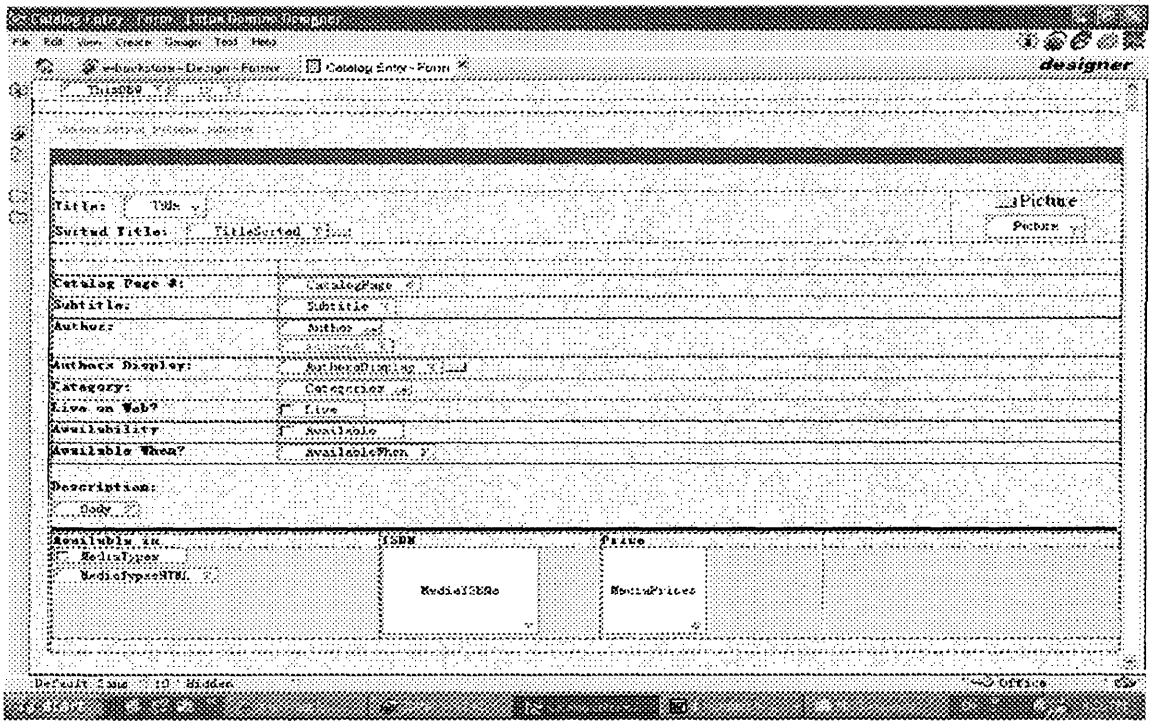


Figure5.10 Catalog Entry form design for client only

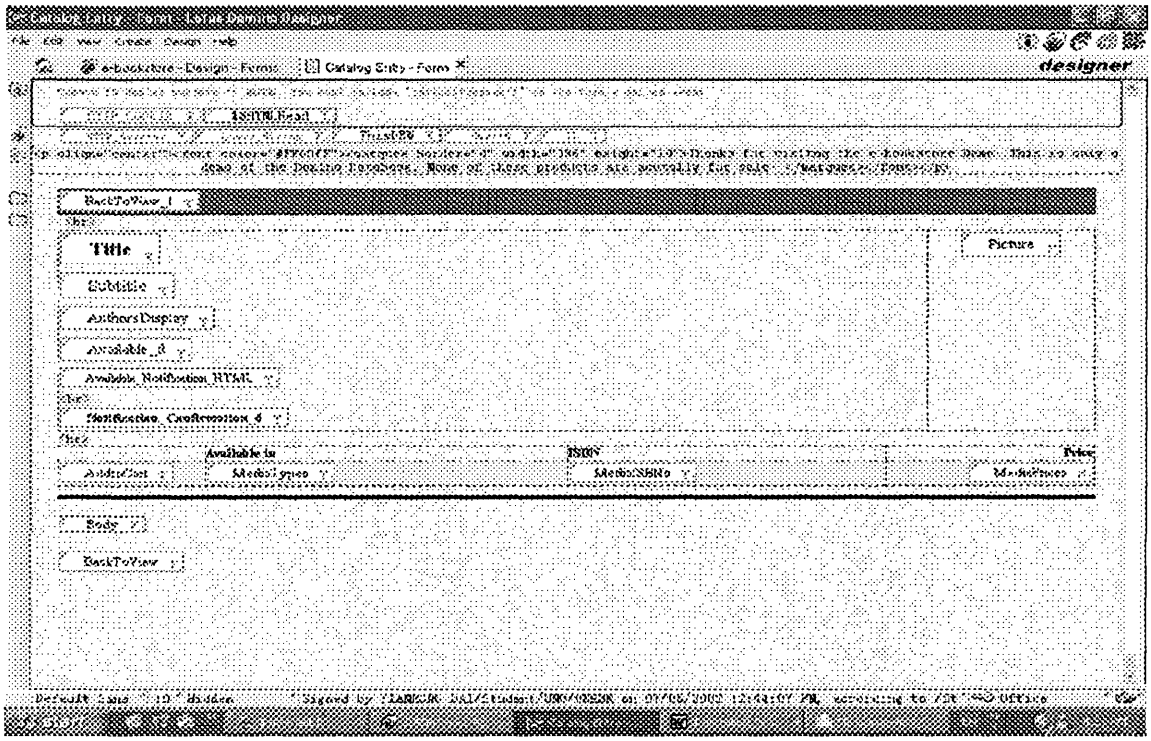


Figure5.11 Catalog Entry form design for web only

5.4 Web User Interface

Since this project purpose was create an e-commerce web site, so I spent most of time on web design, in next section I will talk about what kind of function provide for web users, and how to implement.

5.4.1 Searching Function

As an E-commerce site, this application offers an effective search mechanism. When customers cannot locate products quickly and easily, it won't take them long to find one. Web users can use searching function on every page. It provides five kind of searching key, Title, Description, ISBN, Author and all.

5.4.1.1 Web User Perspective

Jane is visiting E-Bbookstore website, see Figure 5.9 E-Bookstore home page. She is trying to find a book named "JavaScript Sourcebook", but she doesn't know where is this book, so she decides to use searching function. Unfortunately, she even can't remember exactly full book name, so she first click "Title" drop down box, select "Title" as search key, then input "java*" on the bottom left column, and hit "return" key or click "search" button. All the book entries title start with "java" are shown up on the down right column, see Figure 5.10 search results, in this way Jane find her book entry "JavaScript Sourcebook".

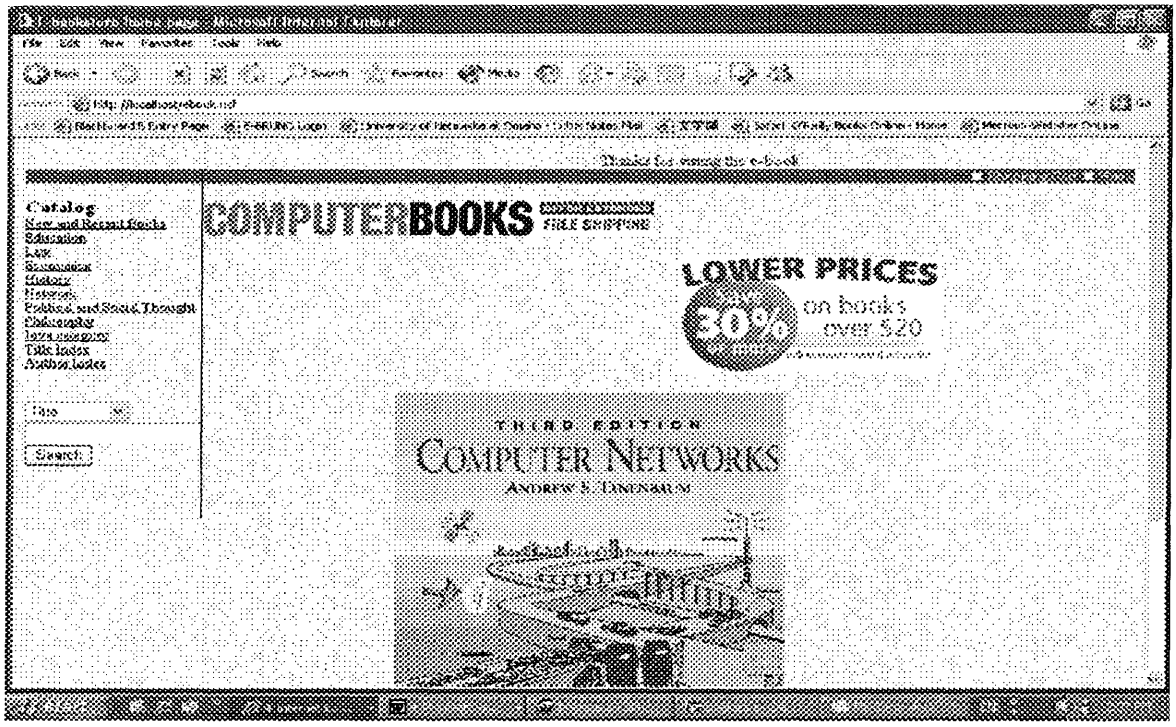


Figure5.12 E-Bookstore Home page

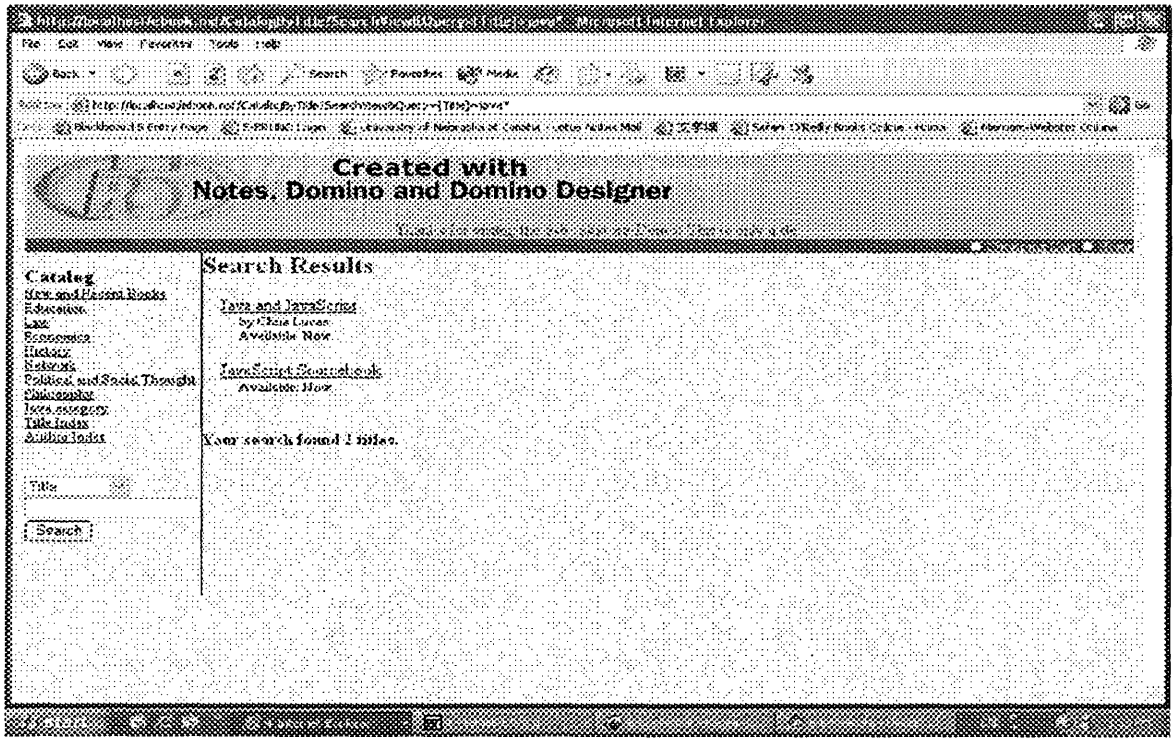


Figure5.13 Search "Java*" results

5.4.1.2 Technical Implementation

This application use a technique called Ad-hoc Queries. With this technique, the user fills out a simple form to define the criteria by which she wishes to filter the data. The form generates a field-specific full-text query, which is then passed back to Domino. Domino in turn returns the results to the user. This technique is an excellent way of letting the user pick a very detailed set of criteria by which to filter the data. And the full-text search engine runs these queries very quickly. Running the query and returning the results to the user involve very little code.

Ideally, the search mechanism should be available anywhere within the site that I choose to include it, keeping the code behind it identical in all instances. Although subforms provide the requisite reusability needed in such a case, an even more flexible option is the use of a shared field. In this case, a shared field called SearchHTML was used to enable the searching capability.

Let's look at the formula behind the SearchHTML shared field:

```
LibraryDBW:=@ReplaceSubstring(@Subset(@DbName;-1);"\\":   "   ";   "/"   :
"+");
"[</FORM><FORM METHOD=post ACTION=\"/" +LibraryDBW +
"/ViewSearchGeneric?CreateDocument\"   ENCTYPE=\"multipart/form-data\">"
+"<BR>" +"<SELECT NAME=\"SearchType\">
<OPTION VALUE=\"Title\" SELECTED>Title
<OPTION VALUE=\"Body\">Description
<OPTION VALUE=\"Author\">Author
<OPTION VALUE=\"MediaISBNs\">ISBN
<OPTION VALUE=\"All\">All</SELECT>
```

```
<br>"+"<INPUT NAME=\"SearchString\" SIZE=\"20\" MAXLENGTH=\"50\">
<br>" +"<INPUT TYPE=Submit Value=\"Search\"></Form>]"
```

Above code function as: Inserts a form in the view which contains a search string field and a find button and which creates a temporary document using the ViewSearchGeneric form based on what was entered in the field. The user never sees the ViewSearchGeneric form.This HTML is generic, it calculates the current database. The sole purpose of this temporary document is to format and pass the search to Domino, then Domino use $$SearchTemplateDefault form to display search results. ViewSearchGeneric form design see figure 5.11, and $$SearchTemplateDefault form design see figure 5.12.

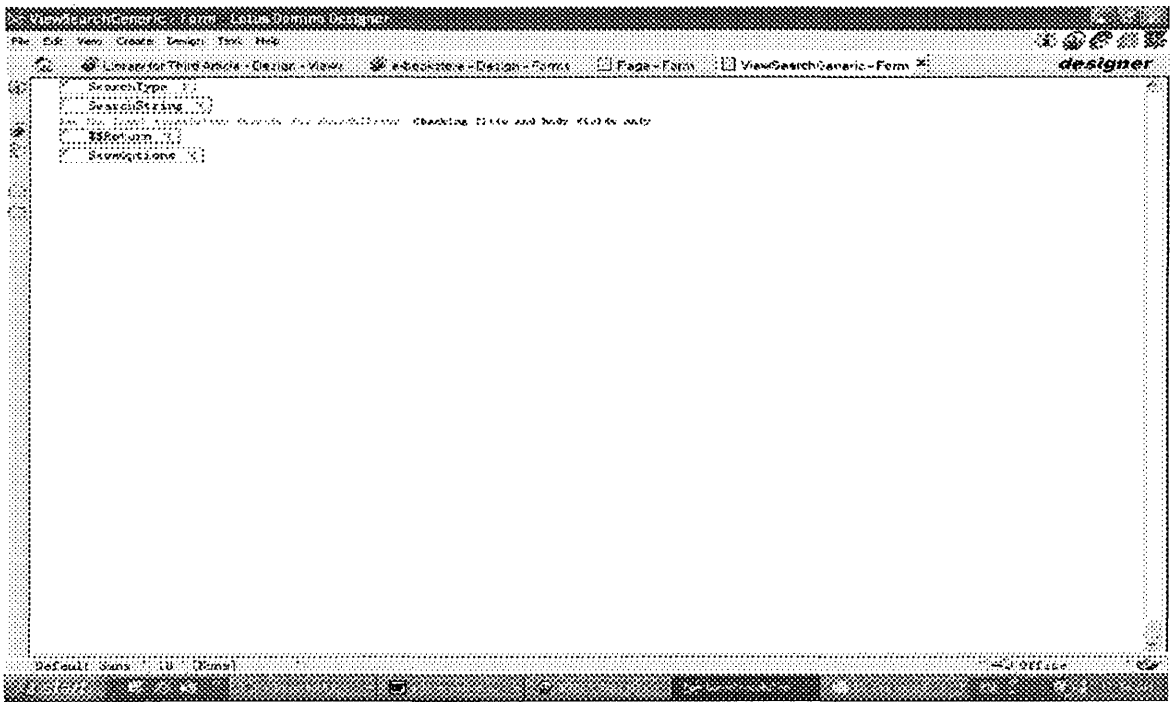


Figure5.14 ViewSearchGeneric form design

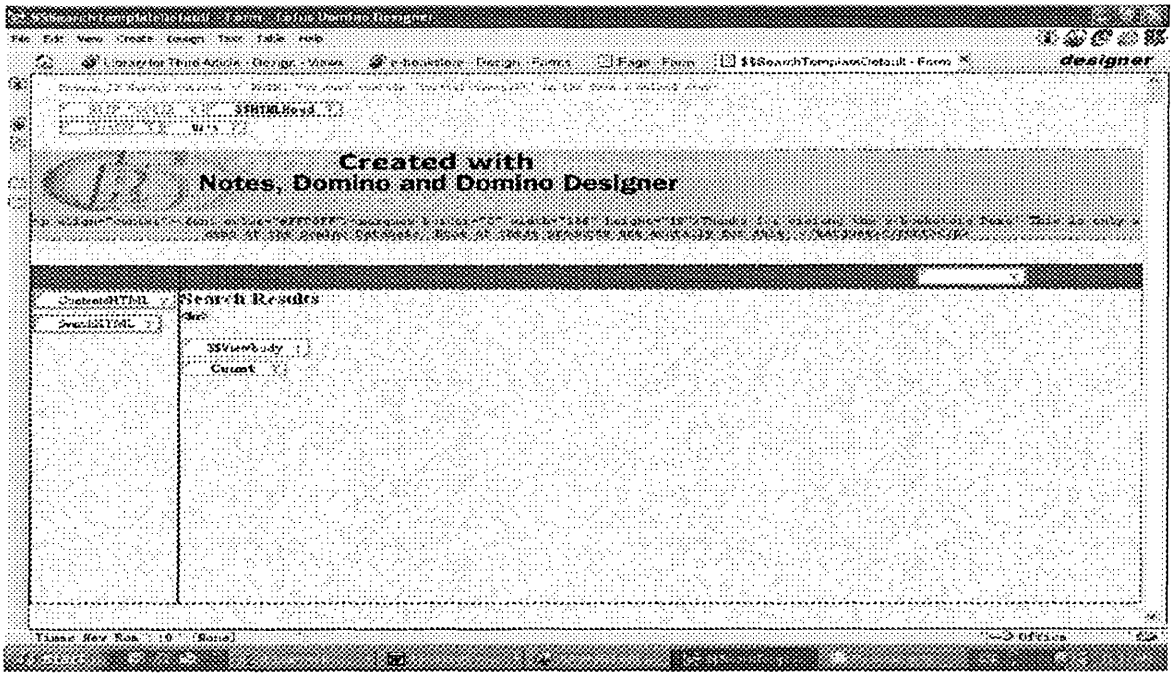


Figure5.15 $$SearchTemplateDefault form design

5.4.2 Dynamic navigation links

A successful E-Commerce site should provide customers with an easy way to browse your catalog. The application not only provide customers clear catalog to browse, but also dynamically add Author Index and Title Index two catalog let customer browse easily.

5.4.2.1 User Perspective

when Jane wants to browse Java book catalog, she click "Java" on left page, all book entries category by "Java" will be displayed on the right below of page, please see Figure 5.13.

There are two more index can be used, one is Title Index, the other one is Author Index. Both of them can be browse using start Letter. For example, Figure 5.14 is the page Jane

get when she click Title Index then click Letter "C" hotlink. Figure 5.15 is the page Jane get when she click Author Index then click Letter "L" hotlink.

5.4.2.2 Technical Implementation

Allowing customers to search for what they want is only one of the crucial components of an e-commerce solution. Providing customers with an easy way to browse your catalog is just as crucial. One of the easiest methods of offering simple navigation throughout an e-commerce site is to identify a limited number of high-level categories by which products can be grouped and then building links to those categories. You can then place these links wherever they are needed, providing a convenient way to hop across product categories regardless of where the customer currently is on the site.

Using shared field

The links are created via the shared field called "ContentsHTML", which is a text field computed for display. Formula behind the field retrieves a hidden view called(NavigatorEntriesHTML), grabs the third column of view, translate to HTML to show up on the web.

Using single-category views

One of R5's great features is the addition of single-category views, which are perfect for displaying categorized product catalogs. As a bonus, they have made life easier on Domino professionals due to their ease of use. In this application, a generic form is used to display a database's multiple single category views.

Let's look at the Catalog View Single Category form (the alias, which the URL references, is ViewCat) design, please see Figure 5.16.

As is typical of Web applications, the form includes an editable, hidden field called Query_String that will capture everything in the current page's URL to the right of the question mark (?). This allows me to parse apart the incoming parameters I have included as part of the URL. Working down the form, you'll notice the computed for display Category_d field immediately above the embedded view. This field's formula will evaluate to the name of the category being displayed:

```
@ReplaceSubstring(@Middle(Query_String  +"&";"&Cat=";"&");  "+";  "  ")
```

Notice that the formula is grabbing everything returned by Query_String that follows &Cat= and that is delimited by another &. Likewise, the embedded view's Embedded selection property extracts its value from the Query_String. Since a view alias is being passed, it's assumed that the alias in this instance doesn't contain spaces, so an @ReplaceSubstring isn't needed:

```
@Middle (Query_String + "&"; "&View="; "&")
```

And, finally, the embedded view's Show single category property can simply point to Category_d since it already has determined the name of the category to be displayed; so it's formula is just Category_d.

WebQueryOpen agent

E-Bookstore wanted the means to browse catalog categories (some of which are quite large) that avoided swamping customers with a single page containing an unmanageable number of items. They also wanted a way to convey to customers their current position

within the category. To that end, a WebQueryOpen agent was incorporated into the forms that display views. This agent calculates the category sequence numbers of the items being displayed as well as seeing to the inclusion of Previous and Next navigation links as needed.

Let's first get a glimpse of what the agent actually does. Figure 5.17 is what Jane sees at the start of the New and Recent Books category.Notice under the category name it says, "Items 1 through 10 of 13" followed by a link labeled, "Last 3 Items."

Take a look at what Jane sees toward the end of the category in Figure 5.18. Now there are two links, one for Previous 10 Items and another for Last 3 Items. This demonstrates that the navigation is indeed dynamic since links are only included when needed; at the beginning of a category there aren't any previous items, so a link to that effect isn't included. Similarly, the Next link changes to a Last link near the end of a category and will disappear altogether when the last item is displayed.

The Catalog View Single Category form's WebQueryOpen object contains the following: formula:

```
@Command([ToolsRunMacro]; "(ViewCat WebQueryOpen)")
```

As the term WebQueryOpen implies, each time this form is called upon by a browser request, it will first execute the specified agent whose purpose is to act upon the page about to be displayed. If you open the sample database in Designer, you'll notice that one of the agents listed is (ViewCat WebQueryOpen). The agent name is surrounded by parentheses because it's a hidden agent, meaning its "When should this agent run?" property is set to "Manually From Agent List." This is of course the agent that performs

all the calculations and builds the links. (ViewBookCatWebQueryOpen)source code see Appendix A.

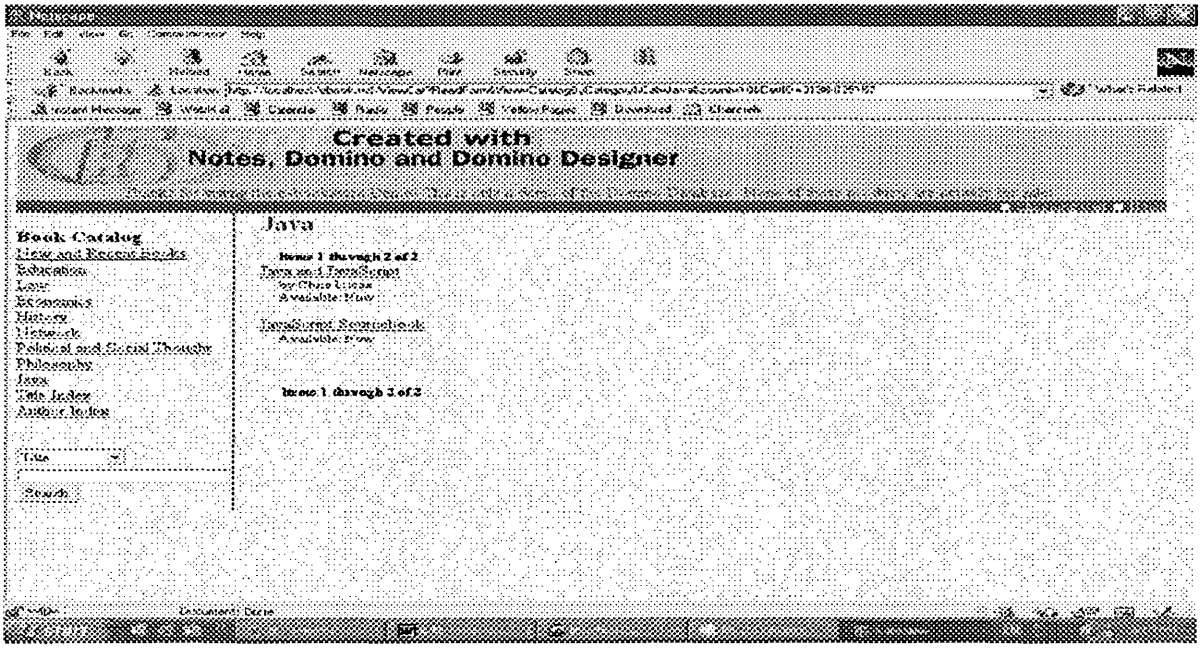


Figure5.16 All book entries category by Java

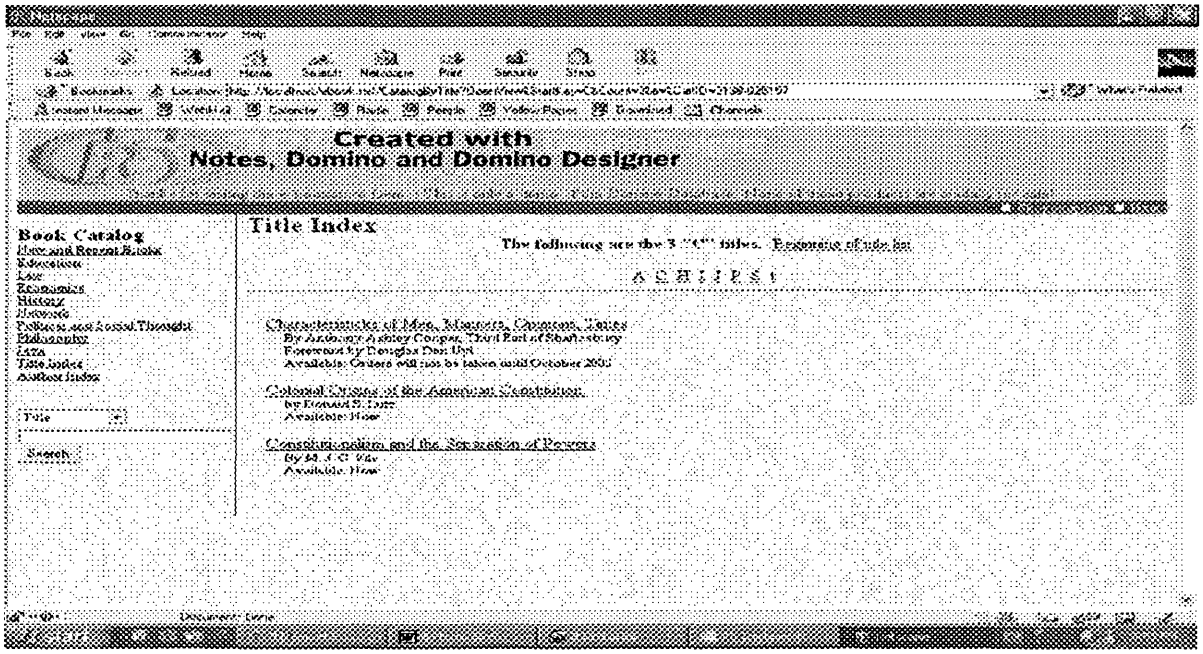


Figure5.17 All book entries start title letter with "C"

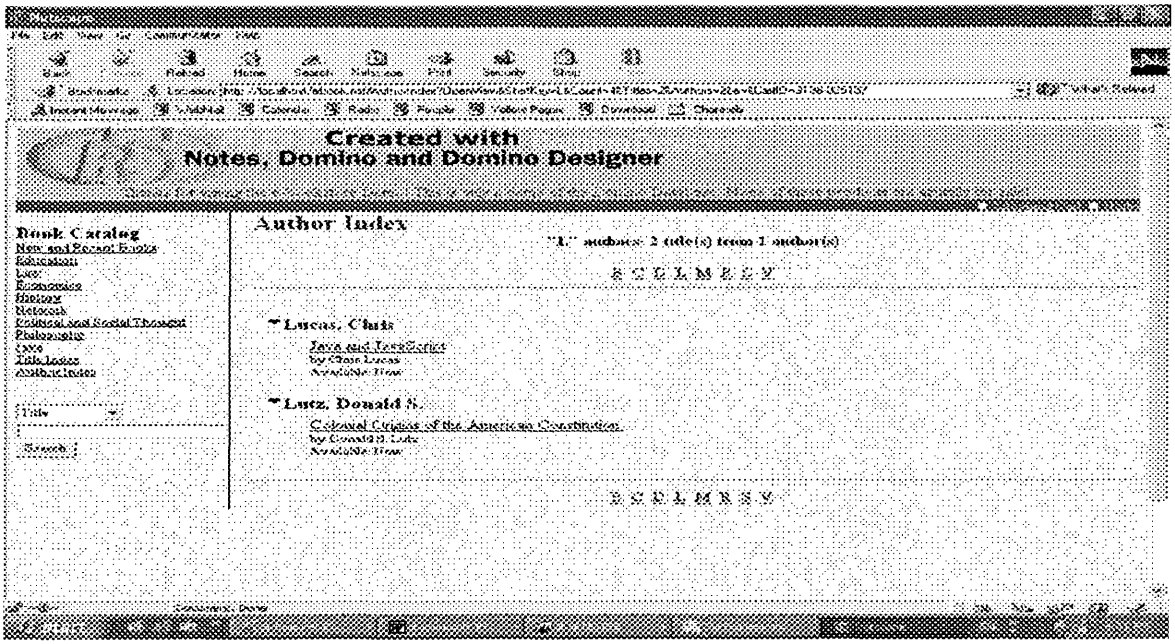


Figure5.18 All book entries start Author letter with "L"

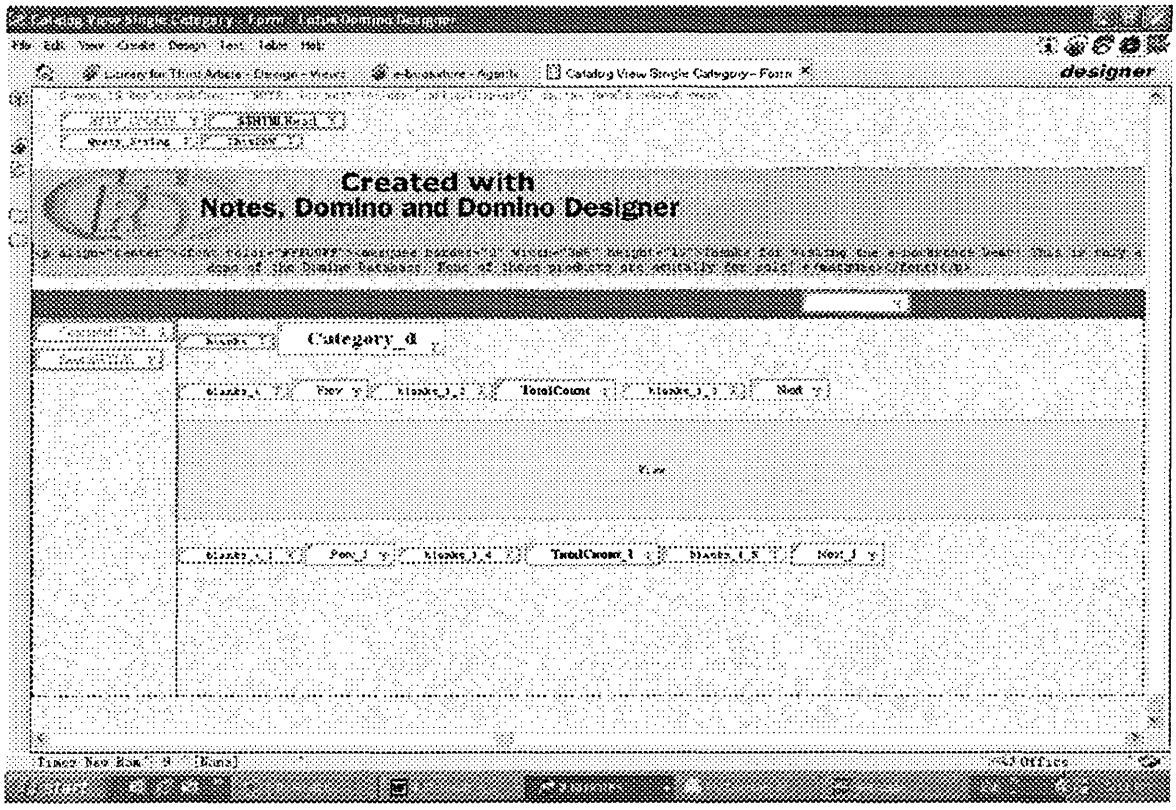


Figure5.19 Catalog View Single Category form design

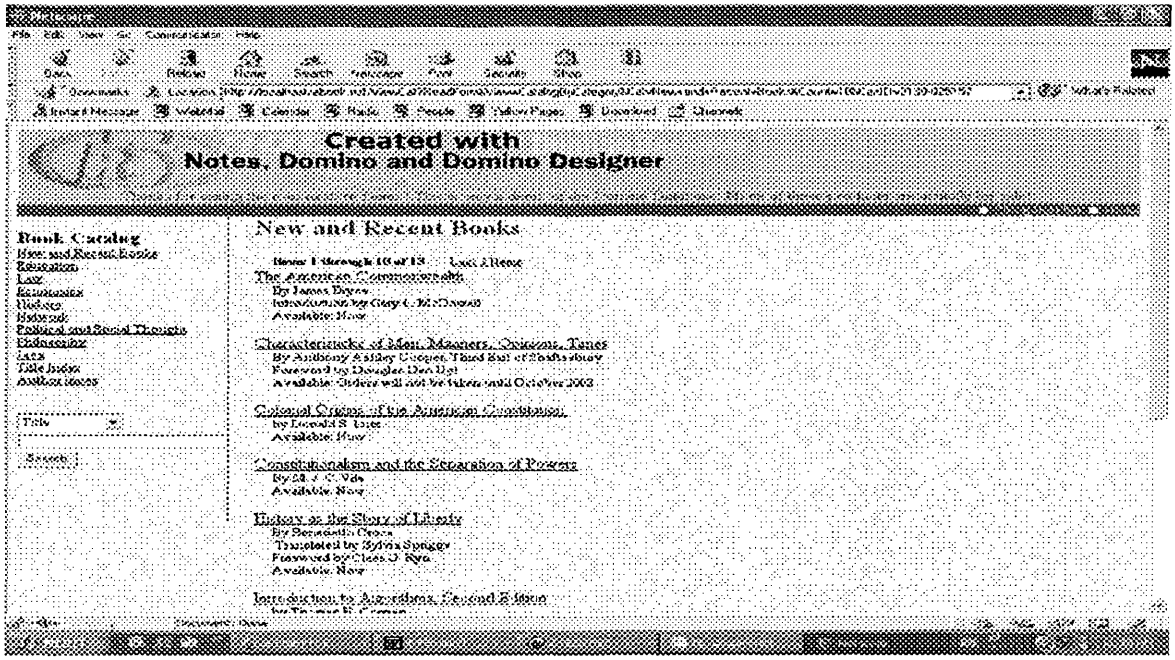


Figure5.20 New and Recent Books Category first 10 book entries

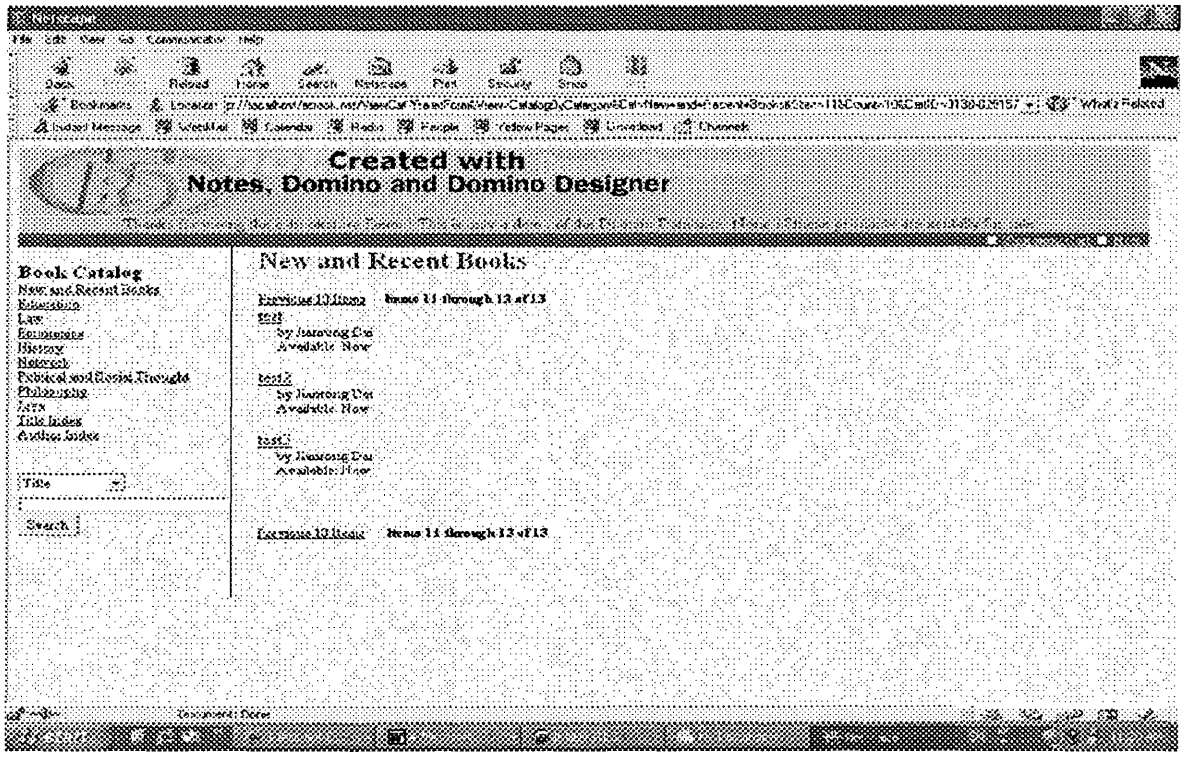


Figure5.21 New and Recent Books Category last 3 book entries

5.4.3 Session Tracking

Web sites are accessed using the HTTP protocol from a Web browser, which is a stateless protocol. If the user interaction involves activity that requires uniqueness such as credentials, or a shopping cart, there needs to be a mechanism to identify and track the user status and information consistently over this stateless channel. This mechanism is commonly known as session control and can be accomplished by such methods as using cookies, URL rewriting.

5.4.3.1 Methods of Session Track

E-bookstore solution makes use of two different techniques to ensure that customers continually identify themselves throughout their sessions.

1. Using cookies

The first technique uses cookies, which are simple to implement and transparent to users. Cookies are small bits of information that servers can place on client machines to, among other things, identify individual users within and across sessions with that same server. In that regard, cookies have "persistence"—they hang around when the browser is shut down and are there to be accessed when it's restarted. For instance, this is what allows you to personalize content at many sites; whenever I return to such a site, they might read a cookie on my machine and from that determine who I am, heralding my return by splashing "Welcome, Jianrong Dai!" across their home page.

In a similar vein, E-bookstore takes advantage of another nice feature of cookies. When a cookie is generated for one of its customers, it is set to expire in one month. That means

that for the next month, a customer is free to visit the site as often as they like, placing items in their shopping cart; and between visits their cart is maintained. When the customer returns, there is no need for them to identify themselves; I simply pull their identification out of the cookie and match it with the carts stored in the database.

So, one option available to me is to generate a unique identifier (usually a number) for each customer and store that value in a cookie on the customer's machine. Once there, each subsequent request will result in the cookie being read, allowing that customer's actions to be tracked. Again, cookies are nice because they are fairly robust. It's difficult (but by no means impossible) for users to tamper with them, and from a user's perspective, they function invisibly.

2. Using JavaScript to append a session ID to links

In the absence of cookies, what other option is available? Once a unique ID has been generated, it can be passed from page to page via the URL. So far so good. It means every link on every page will require the session ID. That's fine for links I'll build "manually" within the code itself, but what about links generated by Domino that I have no control over? Enter JavaScript. With a few short lines of code, I can cycle through all the links on a page (after the server has constructed the page, of course) and append the session ID to each.

5.4.3.2 Session tracking Technique Implementation

The heart of the solution lies with the Common JS Header subform. A subform called JS Head was created. This subform has been included on every form without exception. This

is done for several reasons. First, there is no guarantee as to where a customer will enter the application; if, for instance, a link was mailed to them, they might very well jump straight into a catalog entry, in which case they would still need a session ID just the same as if they had entered the site via the home page. Then there is the case if the URL must transfer the session ID from page to page, that ID must be included as part of every link without fail, thus implying that the mechanism for including it on each link is part of every page/form.Figure 5.19 is the Common JS Header subform in Designer:

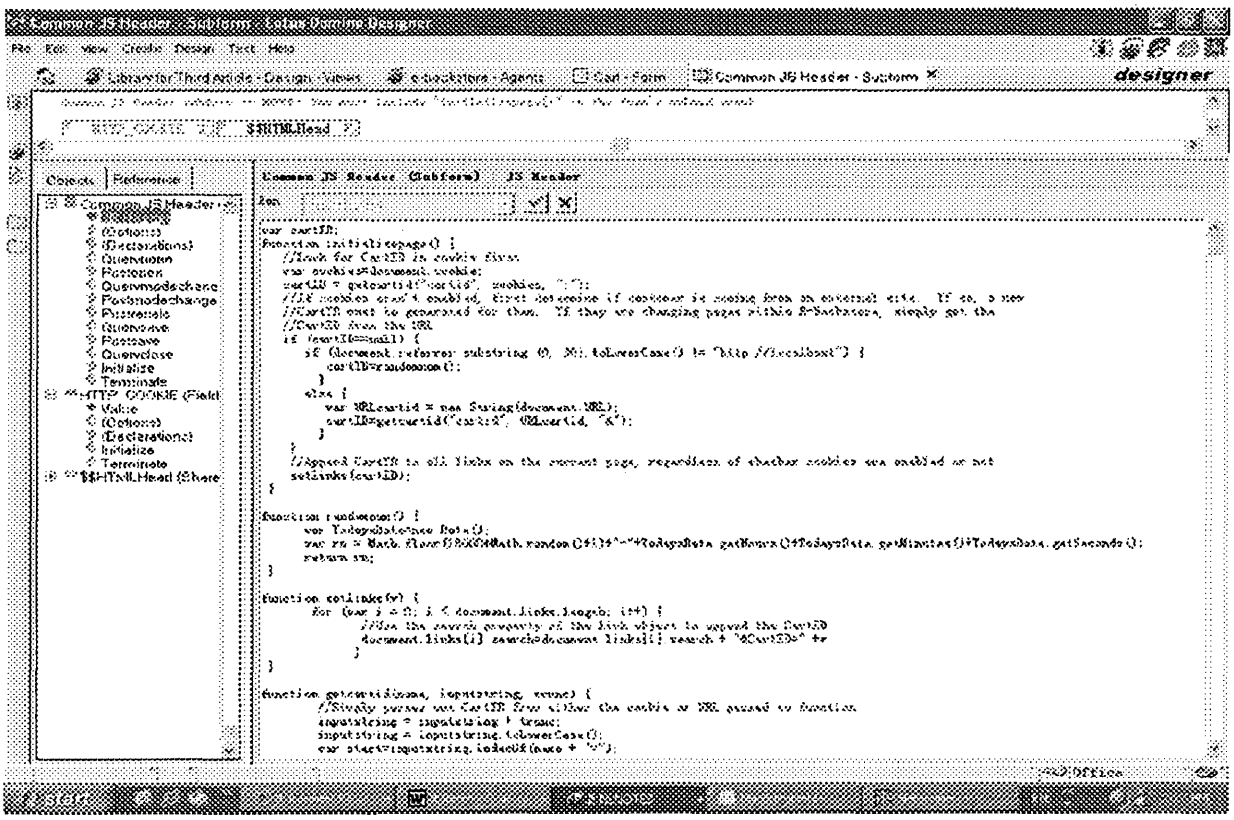


Figure5.22 Common JS Head subform design snapshot

In terms of fields, the subform is pretty simple. It consists of two computed-for-display text fields, one called HTTP_COOKIE and the other, $$HTMLHead. Both of these are marked as "Hide paragraph from Web browsers" in the Field properties box.

HTTP_COOKIE's value is simply set to HTTP_COOKIE, which is a CGI variable automatically returned by the server that contains any cookie information the browser is storing for the current site. $$HTMLHead, as always, is used for any HTML that must precede the form's BODY tag. This is typically where the HTML that creates a cookie is placed. $$HTMLHead formula function as: If cookie already exists for E-bookstore and I won't overwrite it; If no cookie was found, I'll attempt to create one, then write store the cookie on the client.

There is another JavaScript was created for this subform. The variable declarations and functions are all included as part of the subform's JS Header object. Sourece code see Appendix A.

Before examining the functions, it's important to mention that the goal with Javascript code above is to have it execute when each page of the site loads in the browser. Of course, this will only happen when JavaScript is enabled by the browser; when it isn't, I am relying on the cookie to maintain the Cart ID. But JavaScript functions don't call themselves; they must be explicitly executed. This being the case, in conjunction with the Common JS Header subform, each form in the database includes, as part of its onLoad object, the call:

```
initializepage()
```

You'll notice that initializepage is the first function defined in the JavaScript included above. So, whenever any E-Bookstore page loads, initializepage is called. Whether or not anything actually happens is, again, dependent on the JavaScript status of the individual browser. First, initializepage attempts to access any cookies that may exist for this site

through the document.cookie property. If a cookie does exist, the CartID is parsed out via the getcartid function and then it's appended to all the links on the current page.

To summarize, I've just seen two distinctly independent methods that allow browsers to maintain a unique identifier in the form of a CartID. Cookies are preferable, but JavaScript is a capable substitute when cookies are not an option. Either provides Domino with the ability to perform unauthenticated session tracking—the crucial element in any e-commerce solution.

5.4.4 Add-to-Cart Functions

When Jane finds her wanted book "JavaScript Sourcebook", please see Figure 5.20. She wants add to cart first, and check out later, so she click "Add to cart" new book hotlink. She gets Figure 5.21 page. You can notice, she already put an item in the shopping cart. She can continue find other books and put them into cart.

5.4.4.1 Implementation Add-to-cart

Since every customer has been created a unique CartID, and this CartID is kept for one month, I will use this CartID to keep track of customer purchases.

The "Add to cart" link build like this:

"[<A HREF=\"/" + ThisDBW + "/AddtoCart?OpenAgent&ISBN=" + MediaISBNs + "\"><FONT FACE=\"Times New Roman\" COLOR=\"#000080\" SIZE=3 onMouseover=\"this.color=\'#800000\';\" onMouseout=\"this.color=\'#000080\';\">Add to cart</FONT></A>]"

each time "Add to cart" link is clicked, an agent called CartAdd will be called, and also pass a parameter ISBN to this agent. Each link will have one of ISBN number, the CartAdd agent will then use that value to find the item in the database.

Another function of CartAdd agent is creates an "order item" document in the database. This new document stores the CartID of the customer and the specific of item – its ISBN, book type, price, and the quantity being ordered. In the event that an order item document already exists for this customer and item, the quantity will merly be increased by one. So, for each unique item a customer orders, a separate document is created to capture details relating to that item. When CartAdd agent finish all above function, it opens a customer's cart and display the items it contains. Please see Appendix A CartAdd agent source code,

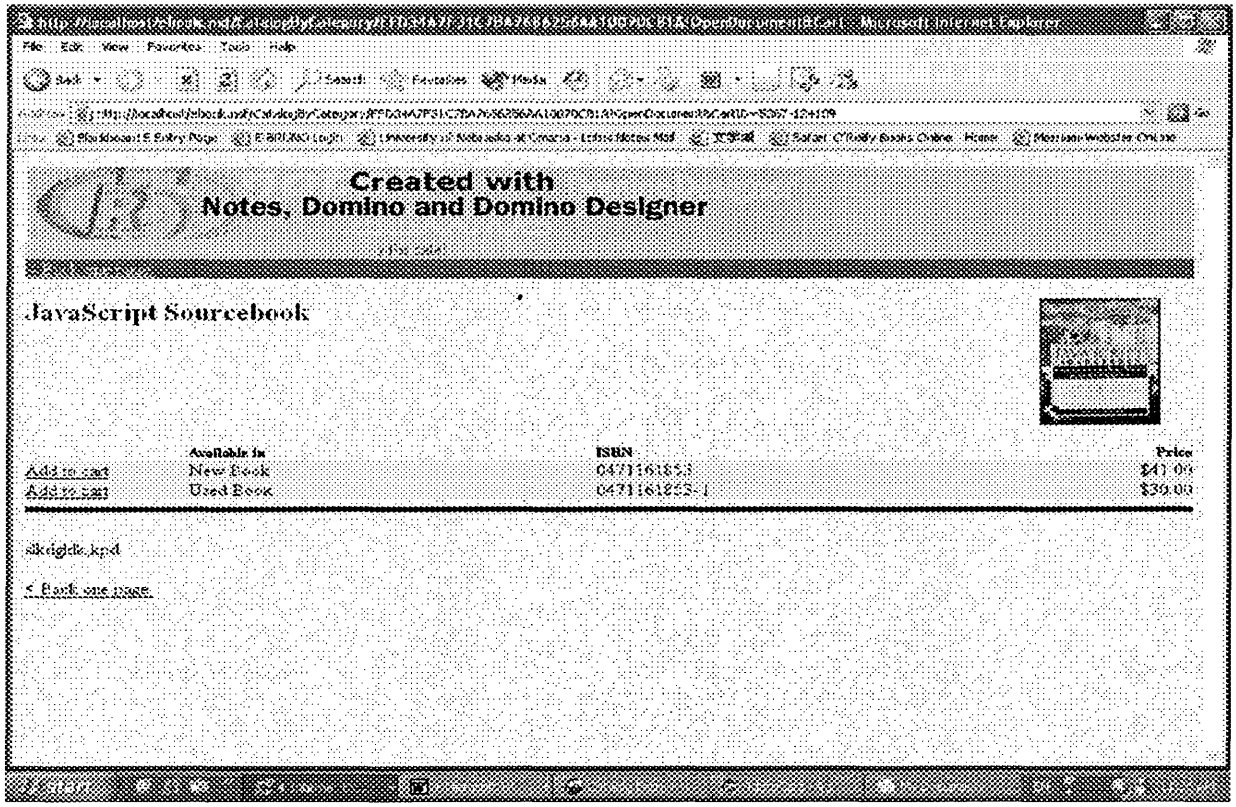


Figure5.23 JavaScript Sourcebook entry web page

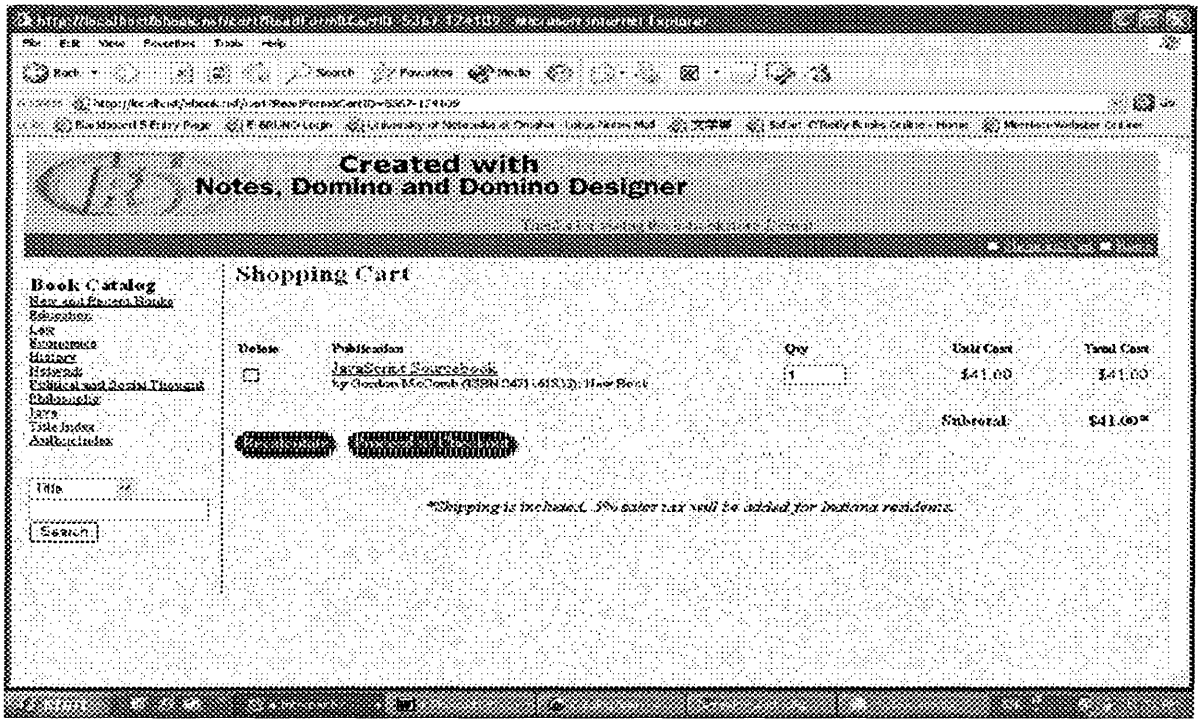


Figure5.24 Shopping cart page

5.4.5 Check Out Scheme

5.4.5.1 User Perspective

Now, Jane has added three items to the cart, now she wants to check out. Wherever she is, only she clicks "Shopping Cart" located right upper of screen, she can review her shopping cart product, see Figure 5.22. She can review each book title, author, ISBN, quantity and total cost. She also can select delete checkbox or edit book quantity then click "Recalculate" button to edit her shopping cart.

Then she clicks "Proceed to Checkout" button to begin check out. Figure 5.23, Figure 5.24 show the screen seen by Jane when she clicks Proceed to Checkout button. The upper portion of the form is where the customer supplies their information. In this

portion, online transaction necessary information such as Credit Card Type, Credit Card Number, Customer Name, Billing Address etc are collected and the bottom shows a summary of the order for verification.

After Jane fill all the information needed, she clicks "Submit your order" button, her order has been saved to database, then, she receives a thank you screen, see Figure 5.25. In the mean time, an auto-confirmation mail will be sent to customer mailbox, see Figure 5.26 for example.

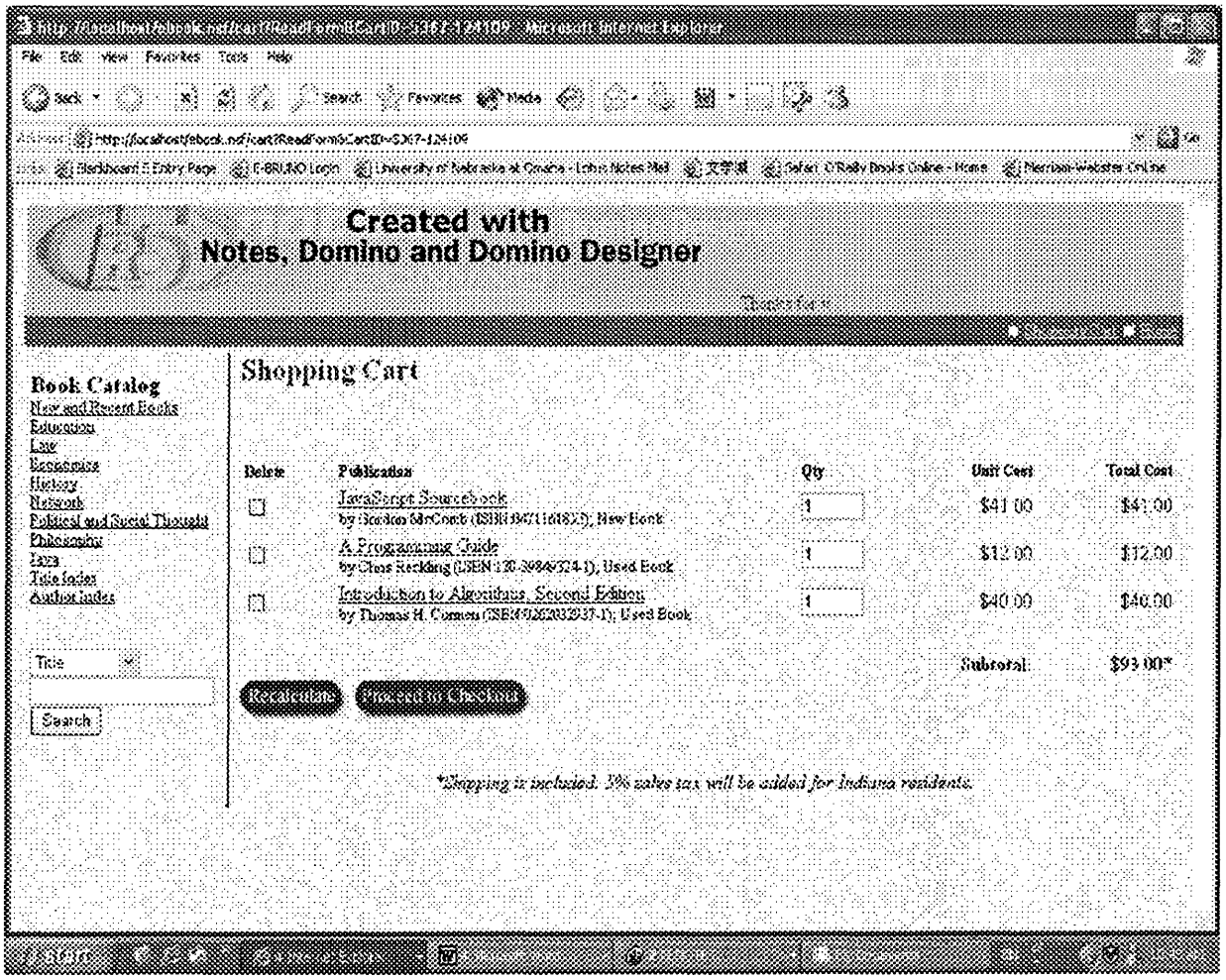


Figure5.25 Shopping cart page snapshot

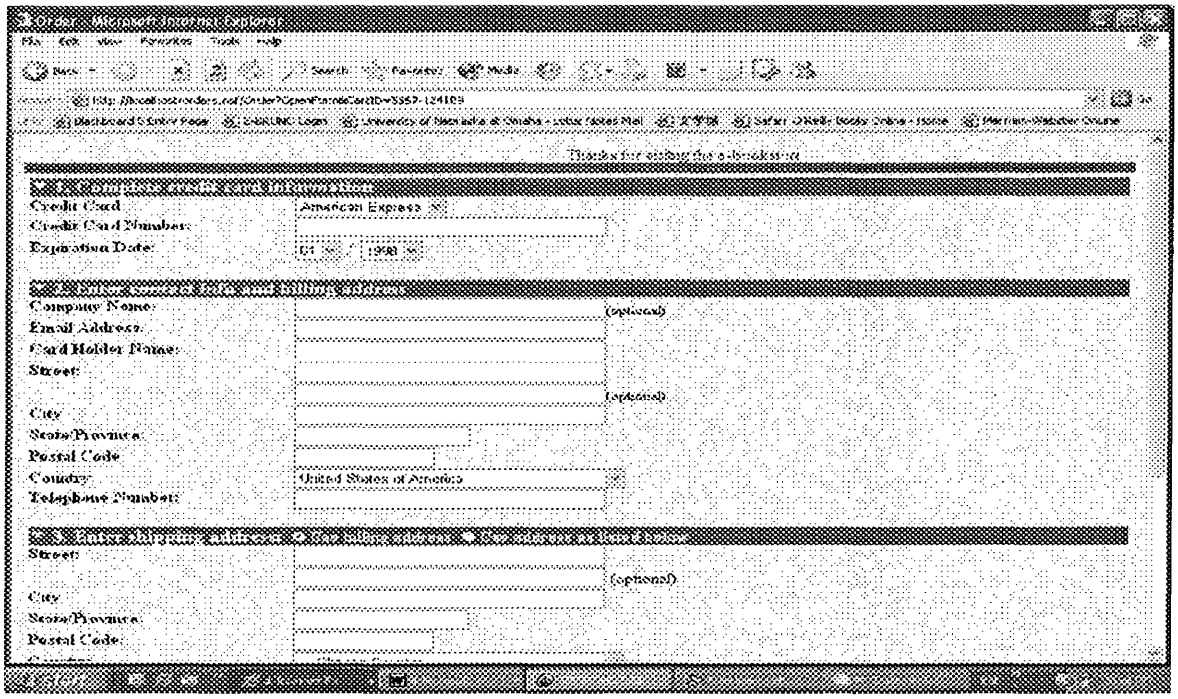


Figure5.26 Order form screen

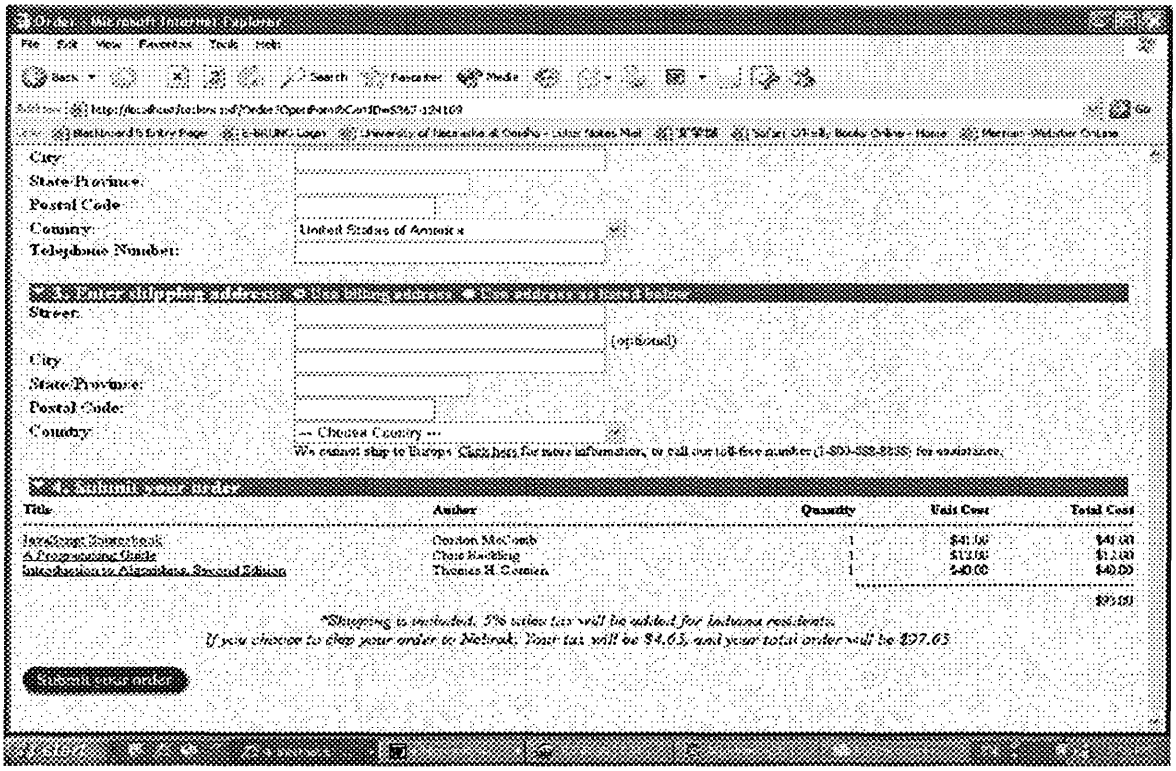


Figure5.27 Order form screen continue

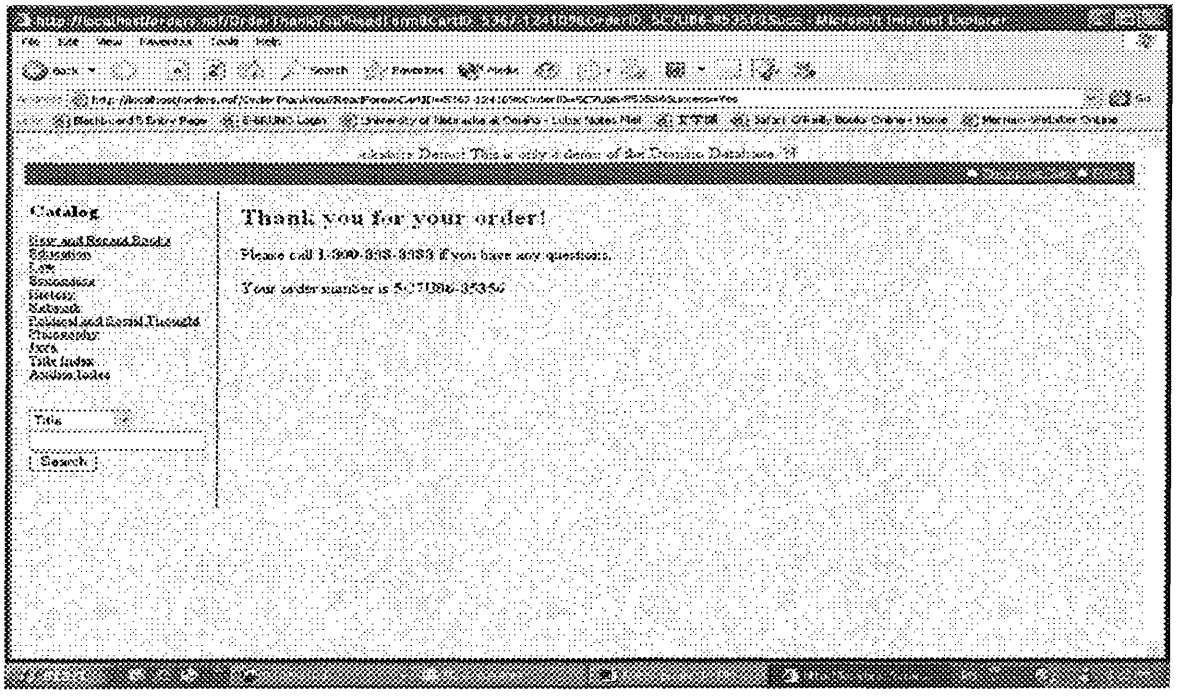


Figure5.28 Thank you page

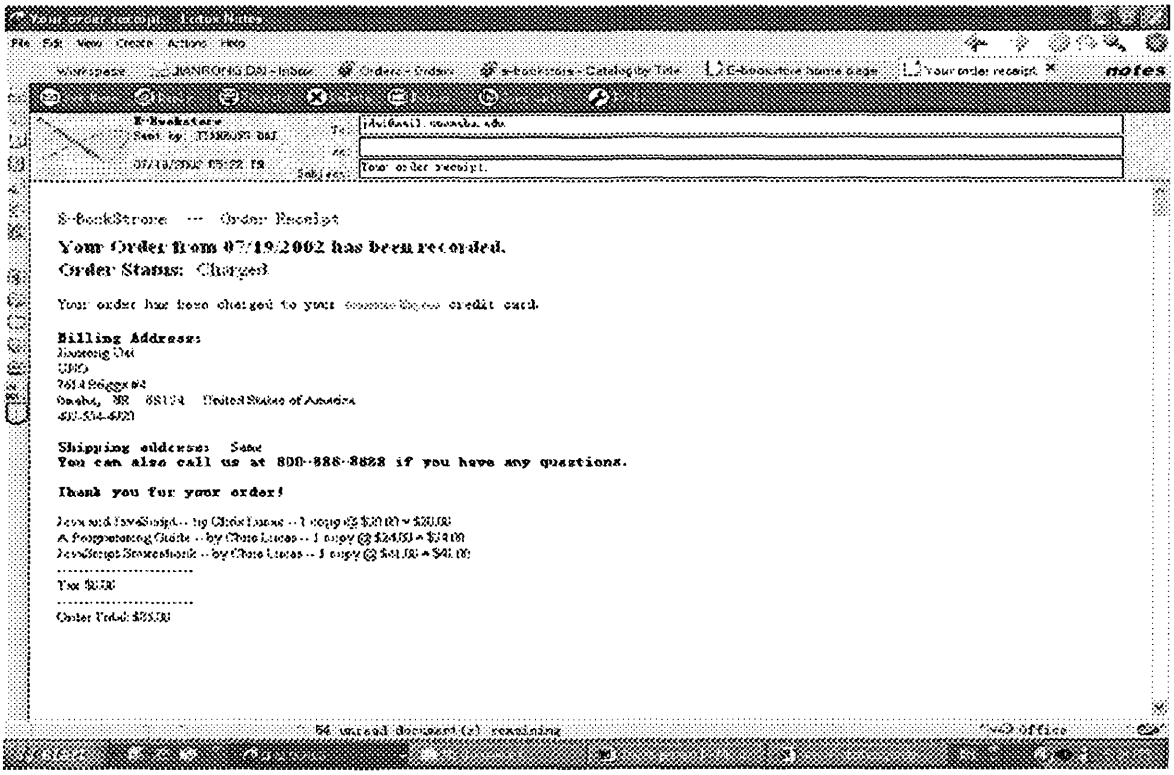


Figure5.29 Confirmation mail snapshot

5.4.5.2 Technical Overview

E-commerce solutions have to be more than just pretty pages, an intuitive shopping cart/checkout scheme is essential for both customer satisfaction and revenue generation—neither of which can live long without the other! First, let's look at all these designs.

Cart Form

The shopping cart is really an aggregation of the entire order item documents created by a given customer. This is accomplished through the use of an embedded single-category view (CartDetailByID), that incorporates both checkboxes and an editable field for the item quantity. Please see Figure 5.23 Cart form in design snapshot.

Its important fields and subform name and function see table 5.3

Table 5.3 Cart Form field and its function

Field Name (column 1)	Function (column 2)
Common JS Header subform	Initialize page, function detail see section 5.3.3.2
Query_String	Capture everything in the current page's URL to the right of the question mark (?)
ThisDBW	Notes database name and path
Server_Name	CGI variable, current host's name
GrabCartID	Retrive the current CartID from a cookie, and if that is not successful(because the customer

	had disabled cookies in their browser), the CartID is parsed out of current page's URL via the Query_String field.
CheckForDocs	Ensure that there are actually items in the current cart to display.
DisplayMessage	Look at checkForDocs and inform the customer of any potential problems.
UserLinks(shared filed)	Build "Shopping Cart" and "Home" link in every page
BeforeView	Povide the cart its column heading- Delete, Publication, Qty, and so on – which you can see in the Figure 5.22
Single-category embedded view	Display those order item documents that match the CatID
AfterView	Calculate the total price

UpdataCart agent

Whenever customer click "Recalculate" button, an agent called UpdataCart is called. The purpose of this agent is simple: update the quantity for each order item in the cart based on customer input, delete those order items that the customer has chosen to remove, and redisplay the updated cart. Please see Appendix A UpdataCart agent source code.

Order Form

When Jane Clicks "Proceed to Checkout", URL is re-directed to another database called orders.nsf. This database is used to store order documents and charge credit etc. Let's see Order Form in designer snapshot in Figure 5.28-5.29. This is a long list form, its important fields and functions see table 5.4

Table 5.4 Order Form Fields and functions

Field Name (column 1)	Function (column 2)
Common JS Header subform	Initialize page, function detail see section 5.3.3.2
Query_String	Capture everything in the current page's URL to the right of the question mark (?)
ThisDBW	Notes database name and path
Server_Name	CGI variable, current host's name
StateCodes	Is muti-value and contains abbreviations for all the states.
StateNames	Contains the full names of the states.
CartID	Capture CartID.
ID	Create a unique ID for each order, it is different from CartID in that it is possible to place multiple orders while still maintain the same CartID.

cr	The created date
Status	Order status(view only by store staff)
Mdump	Help diagnose credit processing errors and is populated by the (CreditCard) agent (view only by store staff).
ErrorMessage	When customer leave any required field blank, it will show error message.
.....	These fields are used to collect credit card, billing and shipping information. There are quite simple, I will not explain them. See Figure 5.29 and top of Figure 5.30
OrderDetail	Contains all the information for the contents of the cart, and subsequent fields on the form refer to it.
ISBNs	Parse all the ISBN values from OrderDetail.
Titles	Parse all the Title values from OrderDetail
Authors	Parse all the Author values from OrderDetail

HotspotTitles	Offer customer a link back to the individual items they're in the process of ordering.
Authors_d	Author Multivalue
Titles_d	Title Multivalue
Qts	Order Quantity
UnitPrices	Unit price
TotalPrices	Qts*UnitPrices
OrderPrice	Sum of TotalPrices
Tax_d	Indiana tax
Tax	Tax fee
OrderPriceTotal	Final price
OrderReceiptDetail	Create a single detail line for each order item to help build an order confirmation e-mail to customer.
OrderReceiptTotalLines	Create total information for the entire order to help build an order confirmation e-mail to customer.
CustFields	All the form fields.
RequiredCustFields	Required fields to Control the field validation performed by

	the(OrderFormWebQuerySave) agent.

OrderFormWebQuerySave agent

When the customer clicks on the "Submit your order" button, an Order document is created in the Orders database. This is, of course, the job of the (OrderFormWebQuerySave) agent. The agent also performs the necessary field validation, and provided the order has passed validation, calls the (CreditCard) agent to actually charge the order. See Appendix A for this agent's complete code. The most important thing to realize with regard to the (OrderFormWebQuerySave) agent is that regardless of whether an order passes field validation or not, an Order document will be saved in the database. This is what enables the customer's previous information to be redisplayed to them. If the order fails validation, it will need to be redisplayed. In other words, the Order document is simply reopened in Edit mode. If, on the other hand, the Order is in good shape, it is saved and the (CreditCard) agent is called.

CreditCard agent

Credit card processing relies on a product called Commerce Accelerator, available from IT Factory. This is the set of LSXs. It interacts with CyberCash (or any similar on-line financial transaction service company), allowing credit card validation and charging to be performed via calls from LotusScript. Note that there are other products available to perform these same functions. The credit card processing in the this application *is not* functional for the simple reason that you would be required to have a valid copy of

Commerce Accelerator installed on your machine in order to successfully complete a transaction.

The agent, however, provides a couple of additional functions. Since it comes at the end of the order process, (CreditCard) agent also mails the customer an e-mail confirmation of their and marks the order item documents in the ebook database with a status that effectively cleans out the customer's cart, CreditCard agent source code see Appendix A. Please see figure 5.31 OrderReceipt designer snapshot and figure 5.32 OrderThank you form designer snapshot.

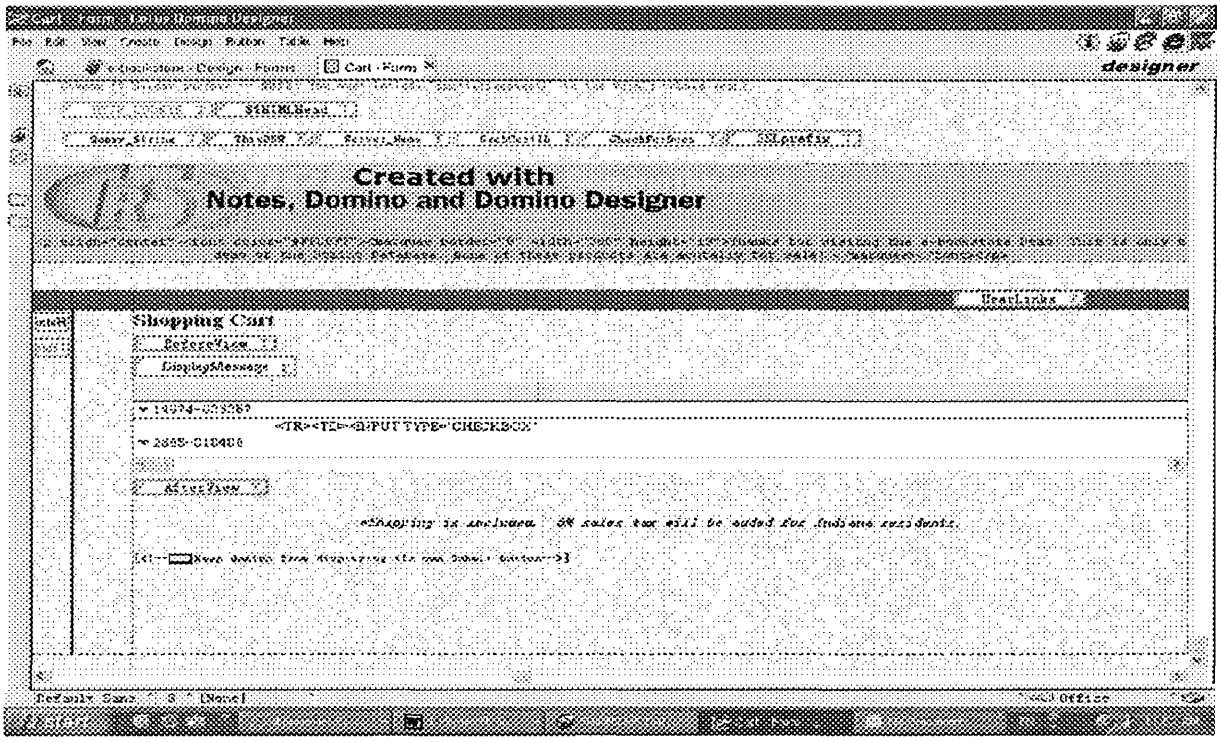


Figure5.30 Cart form in design snapshot

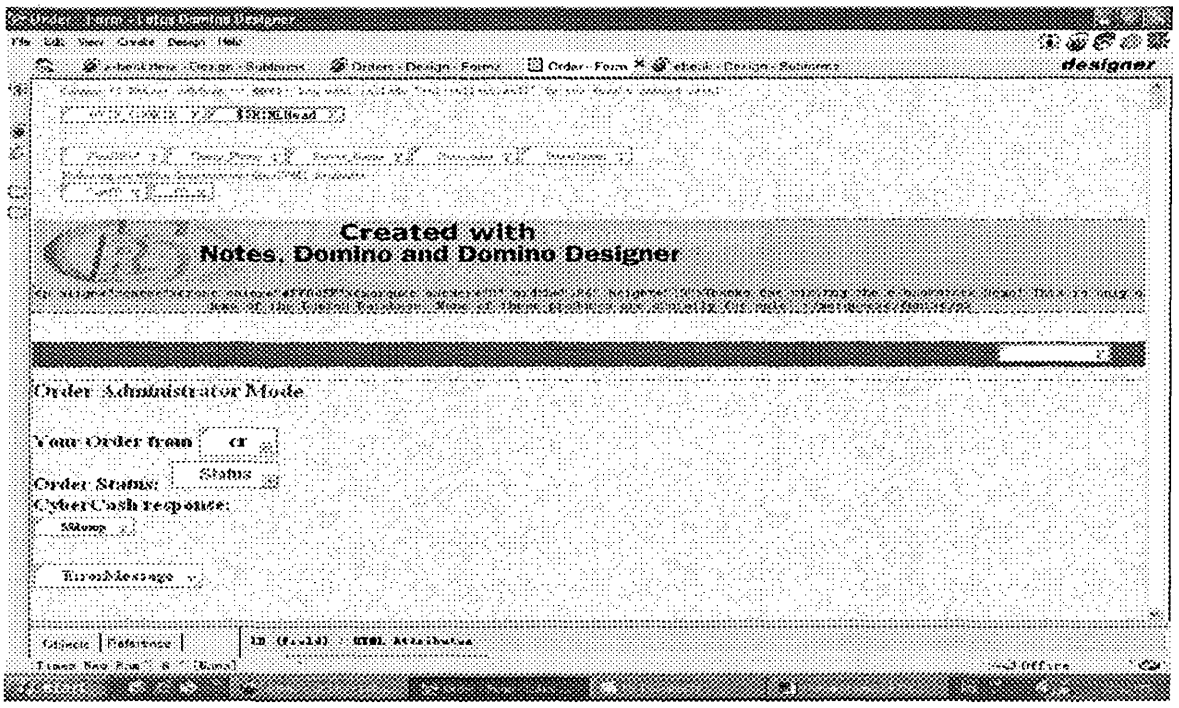


Figure5.31 Order Form design snapshot <1>

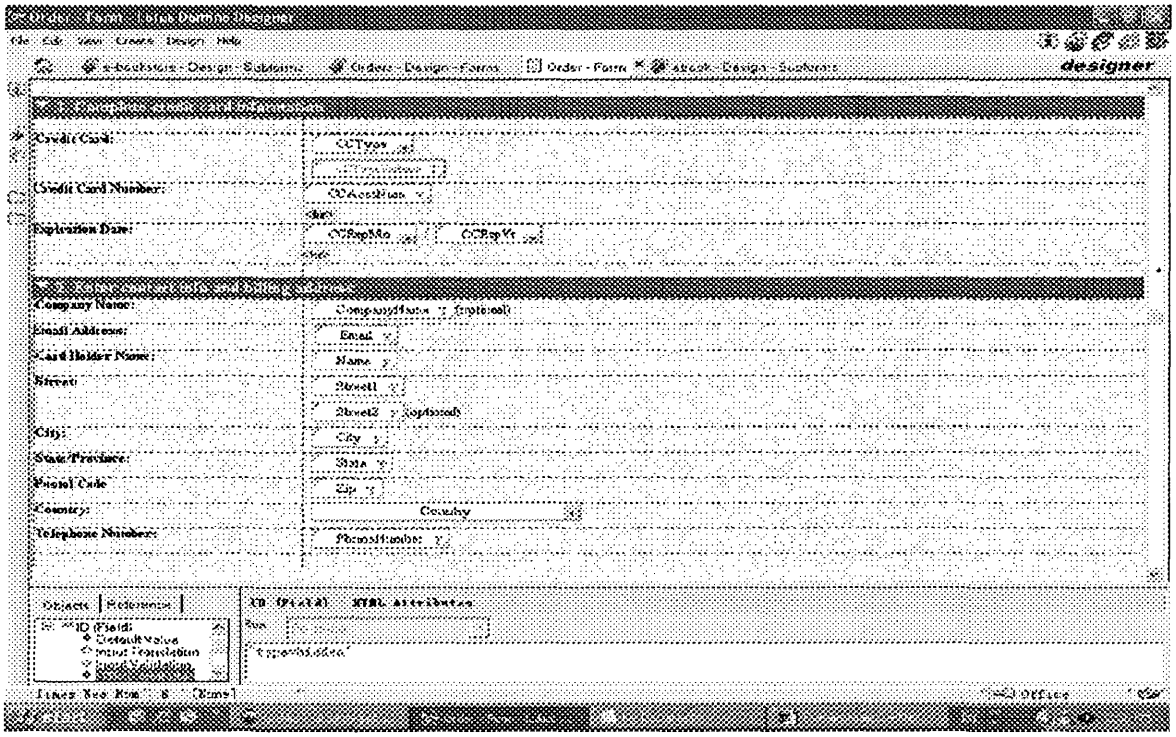


Figure5.32 Order Form design snapshot <2>

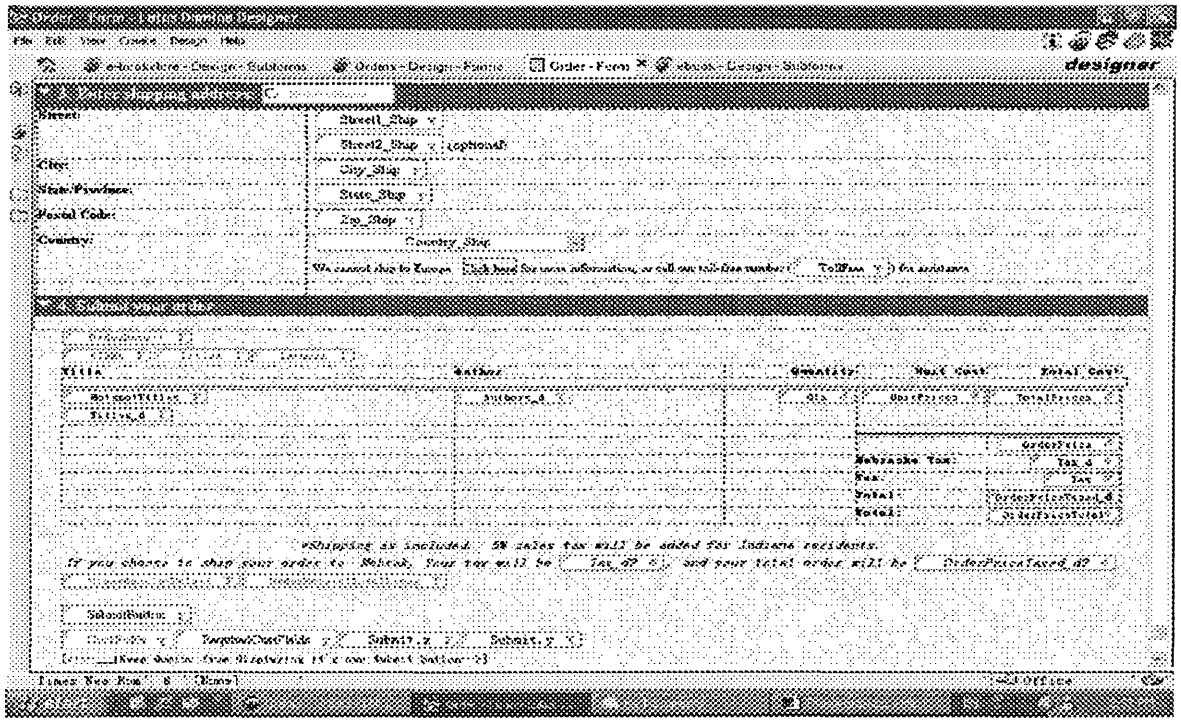


Figure5.33 Order Form design snapshot <3>

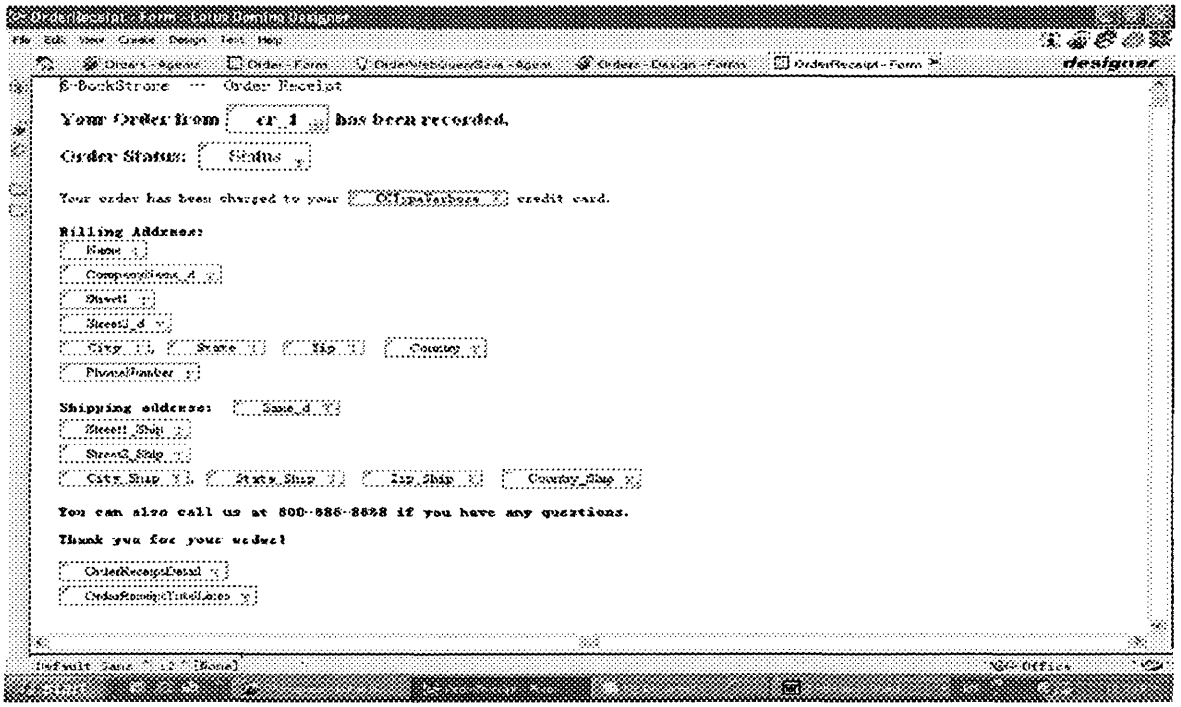


Figure5.34 OrderReceipt designer snapshot

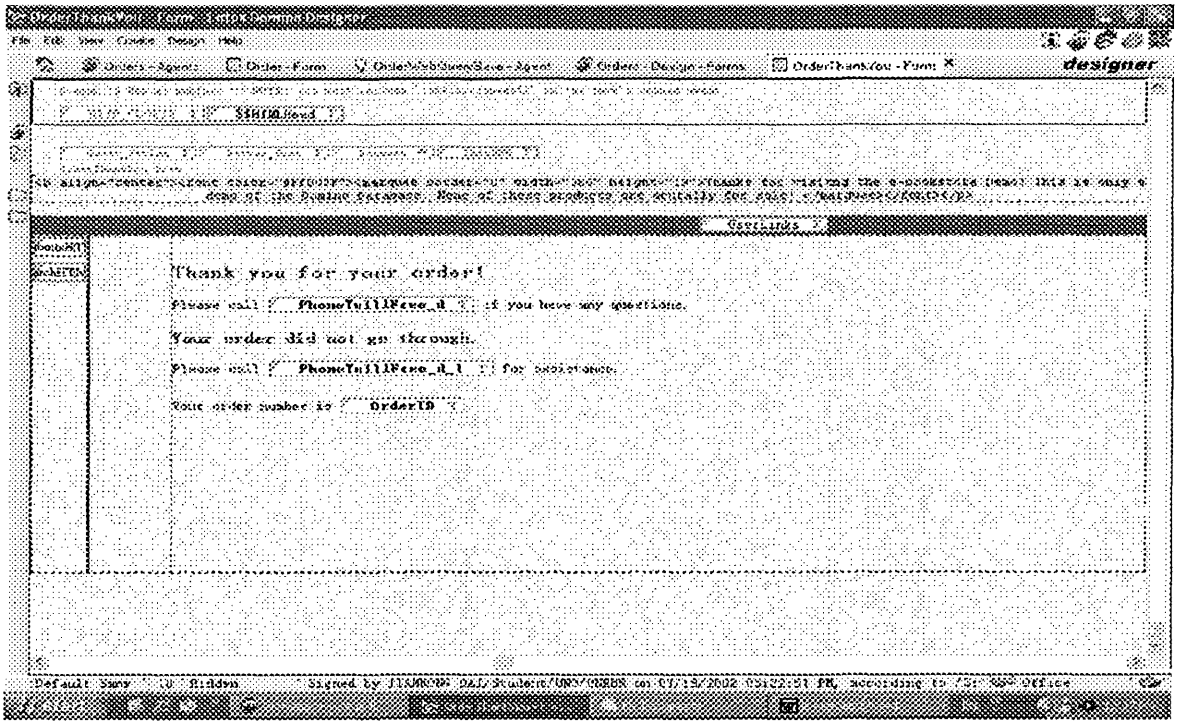


Figure5.35 OrderThank you form designer snapshot

CONCLUSION

This project was to attempt to build a basic functional Domino-powered e-commerce application- E-Bookstore. E-Bookstore web site contains three main components of an e-commerce web site (catalog of items, shopping cart, and checkout function), provides powerful search function to customer. E-Bookstore generates and stores a unique session ID for each web user. The unique session ID is attached to every item a user adds to his shopping cart. This application also provides some back end maintenance functions such as add book category or book entry etc.

Compare to other Successful e-commerce site, I have quite a few features need to polish. For instance, it might be beneficial to store credit card and shipping information so that returning customers are not faced with the rather distasteful chore of re-entering it with every order. Order histories are another useful tool; customers may want a summary of past orders as well as the ability to check the status of their current orders. This extra feature can be extended later.

REFERENCES

1. Violino, Bob., "Building B2B Trust", Computerworld, Vol. 36, No. 25, June 2002, pp. 32-34.
2. Abdur-Razzaq, Baria M., "ELITE E-COMMERCE" , PC Magazine, 08888507, Vol. 21, No. 1, January 2002, pp.26.
3. Dias, D. M.; Palmer, S. L.; Rayfield, J. T., "E-commerce interoperability with IBM's WebSphere Commerce products" , IBM Systems Journal, Vol. 41, No. 2, 2002, pp. 272-286.
4. Benetti, Ilario., "An information integration framework for E-commerce", IEEE Intelligent Systems, Vol. 17, No. 1, January 2002, pp. 18-25.
5. Gordijn, Jaap.; Akkermans, Hans., "Designing and evaluating e-business models", IEEE Intelligent Systems, Vol. 16, No. 4, July/August 2001, pp. 11-17.
6. Standing, Craig., "Methodologies for developing Web applications", Information and Software Technology, Vol. 44, No. 3, March 2002, pp. 151-159.
7. Greenstein, Shane., "E-business infrastructure", IEEE Micro, Vol. 21, No. 6, Nov/Dec. 2001, pp. 70-71.
8. Kowtha, Rao N., "Determinants of website development: a study of electronic commerce in Singapore", Information & Management, Vol. 39, No. 3, December 2002, pp. 227-242.

9. Meyer, Andrew., "E-commerce--an introduction", Computing & Control Engineering Journal, Vol. 11, No. 3, June 2000, pp, 107-108.

10. Smith, R. A., "Trends in e-business technologies", IBM Systems Journal, Vol. 40, No. 1, January 2001, pp. 4-7.

11. Brown, Kyle.; Brown, Kenyon.; Abrahamson, Carl, "Lotus Notes and Domino 5 bible", Publication: Foster City, CA : IDG Books Worldwide, 2000.

12. Kelleher, Rose.; Jones, Tom, "Advanced Domino 5 Web programming", Publication: New York : McGraw-Hill, 1999.

13. Oliver, Steve; Wood, Pete, "Lotus Domino Web site development", Publication: New York : McGraw-Hill, 1998.

14. Tamura, Randy., "Domino 5 Web programming with Java and JavaScript", Publication: Indianapolis, Ind. : Hemel Hempstead : Que ; Prentice Hall, 2000.

15. Patton, Anthony ., "Practical LotusScript", Publication: Greenwich, CT : Manning, 1999.

16. International Business Machines Corporation, "Lotus Notes and Domino R5.0 security infrastructure revealed", Publication: [S.l.] : IBM Corp., 1999.

17. International Business Machines Corporation.; International Technical Support Organization, "Lotus Domino release 5.0 : a developer's handbook" , Publication: Austin, Tex. : IBM International Technical Support Organization, Edition: 2nd ed., 1999.

18. Rochester, Minn., "Developing E-business applications using Lotus Domino for AS/400", Publication: IBM Corp., International Technical Support Organization, 2000.

19. Allotey-Pappoe, Isaac., "Developing an e-business application using Lotus Domino for AS/400", Publication: [S.l.] : IBM Corp., 2000.

20. Lotus Developer Domain, "Notes/Domino", http://www-10.lotus.com/ldd/down.nsf, IBM Notes web site 2003.

21. Randy Tamura, "Lotus Notes and Domino 4.6 Unleashed", Nov 21, 1997.

22. Lotus Developer Domain, "Domino Performance and Scalability: A Guide to Capacity Planning", http://www.lotus.com/developers/itcentralnew.nsf/allpublic/9ED92FB543E1058C852569BA007C48BC?opendocument , IBM Corp., 2000.

APPENDIX

A: The (ViewBookCatWebQueryOpen)agent

Here is the code for the (ViewBookCatWebQueryOpen)agent, it uses LotusScript.

```
Sub Initialize
Dim s As New NotesSession
Dim doc As NotesDocument
Set doc = s.DocumentContext

Call DoPrevNextHTML (doc)

End Sub
Sub DoPrevNextHTML (doc As NotesDocument)
Dim db As NotesDatabase
Dim v As NotesView, entry As NotesViewEntry

Dim intTotal As Integer, intStart As Integer, intCount As Integer, intNextStart As Integer,
intPrevStart As Integer
Dim strCatParm As String, strNextlabel As String, intNextEnd As Integer, strNextParms As String,
strPrevParms As String, strPrevLabel As String, intPrevEnd As Integer
Dim intNext As Integer, intPrev As Integer
Set db = doc.ParentDatabase
Dim vPath As Variant
```

```
vPath=Evaluate({@ReplaceSubstring (@Subset (@DbName; -1); "\\" : " "; "/" : "+")})
vView = Evaluate ({@Middle (Query_String + "&"; "&View="; "&")}, doc)
vCat = Evaluate ({@Replacesubstring (@Middle (Query_String + "&"; "&Cat="; "&"); "+" ; " ")},
doc)
vStart = Evaluate ({@Middle (Query_String + "&"; "&Start="; "&")}, doc)
vCount = Evaluate ({@Middle (Query_String + "&"; "&Count="; "&")}, doc)
If vStart(0) = "" Then
intStart = 1
Else
intStart = Cint (vStart(0))
End If
If vCount(0) = "" Then
intCount = 25
Else
intCount = Cint (vCount(0))
End If
If vCat(0) = "" Then
strCatParm = ""
Else
doc.tempcat = vCat
vCat2 = Evaluate ({ @Replacesubstring (tempcat; " " ; "+") }, doc)
strCatParm = "&Cat=" + vCat2(0)
End If
Set v = db.GetView (vView(0))
If vCat(0) = "" Then
```

```
'doc.PrevNext = "equals blank"
intTotal = v.AllEntries.Count
Else
'doc.prevNext = "else"
Set entry = v.GetEntryByKey(vCat(0))
intTotal = entry.SiblingCount
End If
If intStart = 1 Then
intPrev = False
If intStart + intCount > intTotal Then
intNext = False
Else
intNext = True
intNextStart = intStart + intCount
End If
Else
intPrev = True
If intStart + intCount > intTotal Then
intNext = False
Else
intNext = True
intNextStart = intStart + intCount
End If
intPrevStart = intStart - intCount
If intPrevStart < 1 Then intPrevStart = 1
```

```
End If
strHrefStart = {<A HREF="/} + vPath(0) + {/ViewCat?ReadForm&View=} + vView(0) + strCatParm
strHrefEnd = {</A>}
If intTotal - intNextStart < intCount Then
strNextLabel = "Last " + Cstr (intTotal - intNextStart + 1) + " Items"
Else
strNextLabel = "Next " + Cstr (intCount) + " Items"
End If
strNextParms = {&Start=} + Cstr (intNextStart) + "&Count=" + Cstr(intCount)
If intStart - intCount < 0 Then
strPrevLabel = "First " + Cstr (intCount) + " Items"
Else
strPrevLabel = "Previous " + Cstr (intCount) + " Items"
End If
strPrevParms = {&Start=} + Cstr (intPrevStart) + {&Count=} + Cstr (intCount)
If intPrev Then
doc.Prev = strHrefStart + strPrevParms + {">} + strPrevLabel + strHrefEnd
End If
If intNext Then
doc.Next = strHrefStart + strNextParms + {">} + strNextlabel + strHrefEnd
End If
If (intStart + intCount - 1) > intTotal Then
intShowing = intTotal
Else
intShowing = intStart + intcount - 1
```

```
End If
doc.TotalCount = "Items " + Cstr (intStart) + " through " + Cstr (intShowing) + " of " + Cstr
(intTotal)
doc.Prev_1 = doc.Prev
doc.Next_1 = doc.Next
doc.TotalCount_1 = doc.TotalCount
End Sub
```

B: HS Header Java Script

```
var cartID;
function initializepage() {
  var cookies=document.cookie;
  cartID = getcartid("cartid", cookies, ";");
  if (cartID==null) {
    if (document.referrer.substring (0, 30).toLowerCase() !=
"http://localhost") {
      cartID=randomnum();
      }
    else {
      var URLcartid = new String(document.URL);
      cartID=getcartid("cartid", URLcartid, "&");
      }
```

```
    }
    setlinks(cartID);
}

function randomnum() {
    var TodaysDate=new Date();
    var rn = Math.floor(16000*Math.random()+1)+"-
"+TodaysDate.getHours()+TodaysDate.getMinutes()+TodaysDate.getSeconds();
    return rn;
}

function setlinks(v) {
    for (var i = 0; i < document.links.length; i++) {
        document.links[i].search=document.links[i].search + "&CartID=" +v
        }
}

function getcartid(name, inputstring, trunc) {
    inputstring = inputstring + trunc;
    inputstring = inputstring.toLowerCase();
    var start=inputstring.indexOf(name + "=");
    if (start>-1) {
```

```
        start=inputstring.indexOf("=", start)+1
      }
    var end = inputstring.indexOf(trunc, start);
    if (start==-1 || end==-1) {
        value=null
      }
    else {
        var value=unescape(inputstring.substring(start,end))
      }
    return value;
}
```

C: The CartAdd agent

Here is the code for the CartAdd agent, it is LotusScript:

```
Sub Initialize
Dim s As New NotesSession
Dim db As NotesDatabase
Dim doc As NotesDocument, oiDoc As NotesDocument, cDoc As NotesDocument
Set db = s.CurrentDatabase

Set doc = s.DocumentContext

Dim vCartID As Variant, vISBN As Variant, vItemID As Variant, vMedia As Variant, vQuantity As
```

```
Variant, vPosn As Variant, vPath As Variant
Dim vOrderKey(1) As String
vCartID = Evaluate ({ @Middle (@LowerCase(Query_String) + "&"; "&cartid="; "&") }, doc)
If vCartID(0) = "" Then
vCartID= Evaluate({@Middle(@LowerCase(HTTP_COOKIE) + ";";"cartid=";";")}, doc)
End If
vISBN = Evaluate ({ @Middle (@LowerCase(Query_String) + "&"; "&isbn="; "&") }, doc)
Set cDoc = db.GetView ("ISBNLookup").GetDocumentByKey (vISBN(0))
vOrderKey(0)=vCartID(0)
vOrderKey(1)=vISBN(0)
Set oiDoc = db.GetView ("OrderISBNLookup").GetDocumentByKey (vOrderKey)
If oiDoc Is Nothing Then
Set oiDoc = db.CreateDocument
oiDoc.Form = "OrderItem"
oiDoc.CartID = vCartID
oiDoc.Quantity = 1
oiDoc.ISBN = vISBN
oiDoc.Title = cDoc.Title
oiDoc.Author = cDoc.Author
cDoc.tempISBN = vISBN
vPosn = Evaluate ({@Member (tempISBN; MediaISBNs)}, cDoc) ' position of ISBN in catalog
entry doc
cDoc.tempPosn = vPosn
vMedia = Evaluate ( {@If (tempPosn = 0; "Error"; @Subset (@Subset (MediaTypes; tempPosn); -
1))}, cDoc)
oiDoc.Media = vMedia
```

```
Else
oiDoc.Quantity=oiDoc.Quantity(0) + 1
End If
cDoc.tempISBN = vISBN
vPosn = Evaluate ({@Member (tempISBN; @lowercase (MediaISBNs))}, cDoc) ' position of ISBN
in catalog entry doc
cDoc.tempPosn = vPosn
vPrice = Evaluate ( {@If (tempPosn = 0; "Error"; @Subset (@Subset (MediaPrices; tempPosn); -
1))}, cDoc)
oiDoc.Price = vPrice

Call oiDoc.Save (True, True)
vPath=Evaluate({@ReplaceSubstring (@Subset (@DbName; -1); "\\" : " "; "/" : "+")})
Print "[" + vPath(0) + "/cart?ReadForm&CartID=" + vCartID(0) + "]"

End Sub
```

D: The EditCart agent

Here is the code for the EditCart agent:

```
Sub Initialize
Dim s As New NotesSession, db As NotesDatabase
Dim doc As NotesDocument, oiDoc As NotesDocument

Set db = s.CurrentDatabase
Set doc = s.DocumentContext
```

```
    Dim vCartID As Variant, vPath As Variant, vContent As Variant
    Dim vDelete As Variant, vQuantity As Variant, vISBN As Variant, vSessionID As Variant
    Dim emptyQuantity As Integer

    vCartID= Evaluate({@Middle(@LowerCase(HTTP_COOKIE) + ";";"cartid=";";")}, doc)
    If vCartID(0) = "" Then
            vCartID = Evaluate ({@Middle (@LowerCase(HTTP_Referer) + "&"; "&cartid=";   "&")},
doc)
    End If

    vContent = Evaluate({@Explode(Request_Content;"&")}, doc

    Forall arg In vContent
            If emptyQuantity Then
                    arg = "ISBN="
                    emptyQuantity = False
            End If
            If arg = "QUANTITY=" Then
                    emptyQuantity = True
            End If
    End Forall

    doc.tmpContent = vContent
    vDelete=Evaluate({@Trim(@Right(tmpContent;"DELETE="))},doc)
    vQuantity=Evaluate({@Trim(@Right(tmpContent;"QUANTITY="))},doc)
```

```
vISBN=Evaluate({@Trim(@Right(tmpContent;"ISBN="))},doc)
For x=0 To Ubound(vISBN)
        Set oidoc = db.GetDocumentByUNID(vISBN(x))
        If Not oidoc Is Nothing Then
                If Cint(vQuantity(x))>0 Then
                        oidoc.Quantity = Cint(vQuantity(x))
                        Call oidoc.Save(True,True)
                End If
        End If
Next

If Not vDelete(0) ="" Then
        Forall v In vDelete
                Set oidoc = db.GetDocumentByUNID(v)
                If Not oidoc Is Nothing Then
                        Call oidoc.Remove(True)
                End If
        End Forall
End If

vPath=Evaluate({@ReplaceSubstring (@Subset (@DbName; -1); "\\" : " "; "/" : "+")})
Print "[" + vPath(0) + "/cart?ReadForm&CartID=" + vCartID(0) + "]"

End Sub
```

E: The OrderFormWebQuerySave agent

Here is the code for the OrderFormWebQuerySave agent:

```
Sub Initialize
Dim s As New NotesSession, db As NotesDatabase
Dim doc As NotesDocument
Dim strOrderID As String, strCartID As String
Dim strDBPath As String, strErrorMessage As String, vCartID As Variant

Set db = s.CurrentDatabase
Set doc=s.DocumentContext

Dim vPath As Variant
vPath=Evaluate({@ReplaceSubstring (@Subset (@DbName; -1); "\\" : " "; "/" : "+")})
strCartID = doc.CartID(0)
strOrderID = doc.ID(0)
Dim vFieldVal As Variant
Forall ReqField In doc.RequiredCustFields
vFieldval = doc.GetItemValue (ReqField)
If vFieldval(0) = "" Then
doc.Status = "Draft"
Call doc.Save (False, True)
Goto ReturnErrorMessage
End If
End Forall
doc.Status = "ReadyToCharge"
Call doc.Save (False, True)
```

```
Print "[/" + vPath(0) + "/CreditCard?OpenAgent&CartID=" + strCartID + "&r=" + strOrderID + "]"
Exit Sub
ReturnErrorMessage:
Print "[/" + vPath(0) + "/0/" + doc.UniversalID + "?EditDocument&CartID=" + strCartID +
"&ValError]"
End Sub
```

F: The CreditCard agent

Here is the code for the CreditCard agent:

```
Sub Initialize
Dim s As New NotesSession, db As NotesDatabase
Dim doc As NotesDocument, odoc As NotesDocument

Set doc = s.DocumentContext
Set db = doc.ParentDatabase
Dim vCartID As Variant, vSessionID As Variant, vOrderID As Variant

Dim vPath As Variant


vPath=Evaluate({@ReplaceSubstring (@Subset (@DbName; -1); "\\" : " "; "/" : "+")})

vCartID = Evaluate ( {@Middle (Query_String + "&"; "&CartID="; "&") }, doc)
vOrderID = Evaluate ( {@Middle (Query_String + "&"; "&r="; "&") }, doc)

Set odoc = db.GetView ("(OrdersByID)").GetDocumentByKey (vOrderID(0), True)
```

```
Dim vAmount As Variant, strParm As String

If odoc.Mstatus(0) = "success" Then
odoc.Status = "Charged"
Call SendReceipt (s, db, odoc)
Call ClearCart (s, db, odoc)
strParm = "Yes"
Else
odoc.Status = "ChargeFailed"
strParm = "No"
End If

Call odoc.save (False, True)

Print "[" + vpath(0) + "/OrderThankYou?ReadForm&CartID=" + vCartID(0) + "&OrderID=" +
vOrderID(0) + "&Success=" + strParm + "]"

End Sub

Sub SendReceipt (s As NotesSession, db As NotesDatabase, odoc As NotesDocument)

Dim v As NotesView

Dim cdoc As NotesDocument, mdoc As NotesDocument
Set mDoc = db.CreateDocument
```

```
mDoc.Form = "Memo"
mDoc.Principal = "Catalog"
mDoc.SendTo =odoc.Email
mDoc.Subject = "Your order receipt."

Dim rt As New NotesRichTextItem (mdoc, "Body")
odoc.Form = "OrderReceipt"
Call odoc.Save (False, True)
Call odoc.RenderToRTItem ( rt )
odoc.Form = "Order"
Call odoc.RemoveItem ("OrdersDBW")
Call odoc.Save (False, True)
Call mDoc.Send (False)
End Sub

Sub ClearCart (s As NotesSession, db As NotesDatabase, odoc As NotesDocument)
Dim CartColl As NotesDocumentCollection
Dim LibraryDB As NotesDatabase
Dim strCartID As String
Dim strDBPath As String

strDBPath=db.GetProfileDocument("ApplicationSettings").LibraryDB(0)
Set LibraryDB = New NotesDatabase(db.Server,strDBPath)
Set CartColl = LibraryDB.GetView ("(CartDetailByID)").GetAllDocumentsByKey (odoc.CartID(0))
Call CartColl.StampAll ("Status", "Ordered")
Call CartColl.StampAll ("OrderID", odoc.ID(0))
```

```
End Sub
```