Three Integration Methods for a Component-based NetPay Vendor System

Xiaoling Dai[1] and John Grundy[2]

Department of Mathematics and Computing Science
The University of the South Pacific, Laucala Campus, Suva, Fiji[1]
dai_s@usp.ac.fj

Department of Electrical and Computer Engineering and Department of Computer Science[2]
University of Auckland, Private Bag 92019, Auckland, New Zealand
john-g@cs.auckland.ac.nz

Abstract. We have developed NetPay, a micro-payment protocol characterized by off-line processing, customer anonymity and relatively high performance and security using one-way hashing functions for encryption. In our NetPay prototypes we have designed and implemented two kinds of NetPay vendor systems which use thin-client user interfaces, a component-based server-side infrastructure and CORBA remote objects for inter-system communications. We describe three alternative ways to integrate NetPay interface facilities into these existing E-Journal web applications. We describe the relative strengths and weaknesses with each approach and our experiences building prototypes of them.

Key words: Electronic commerce; Micro-payment; System integration

1 Introduction

World-wide proliferation of the Internet led to the birth of electronic commerce, a business environment that allows the transfer of electronic payments as well as transactional information via the Internet. However, the problem of paying for large-volume, small-value items of information is still to be solved. We have developed a new micro-payment protocol called NetPay [1]. The NetPay protocol allows customers to buy E-coins, worth very small amounts of money, from a broker and spend these E-coins at various vendor sites to pay for large numbers of discrete information or services of small value each. There are a number of other micro-payment systems such as PayWord [5], Millicent [4], and PayFair [6]. However, most existing or proposed micro-payment technologies suffer from problems with communication, security, lack of anonymity and being overly vendor-specific. We have developed the NetPay protocol and supporting architecture to address problems with communication, security, anonymity and vendor-specific. In this paper, we give an overview of existing micro-payment models and point out the problems with these models. We then briefly describe our

CORBA-based and component-based NetPay micro-payment system design. We then describe the designs for the three ways to integrate NetPay interface facilities into an existing web application. We conclude with an outline of our further plans for research and development in this area.

2 CORBA-based NetPay Architecture

We initially developed a software architecture for implementing NetPay-based micro-payment systems for thin-client web applications that used hard-coded vendor facilities for micro-payments [2]. NetPay additions to the web components included pricing for items, pay-per-click for each item, a server-side e-coins spent database, and nightly redemption of spent e-coins with a broker in exchange for real money. The existing web server components' code was modified to interact with the NetPay functions when needed. There are some major disadvantages to this approach:

- It is difficult and time consuming to add NetPay support to existing applications.
- The reusability level is lower. For example, when some NetPay functions and interfaces are fitted into an E-newspaper system developers can't reuse the same NetPay objects into another existing web application without modification.
- Enhancement of the NetPay functions means modifying the web components.

To overcome these disadvantages, a component-based NetPay vendor system was developed. One of the characteristics of such a system is that its components can be plugged into an existing web system. The new system should match the needs of existing systems and have NetPay micro-payment support integrated seamlessly with minimum effort with their existing web application architecture. We need to enhance the existing, domain-specific components of an existing NetPay vendor system and, using plug-and-play, add the NetPay components to the existing web application. In an E-journal example, the journal provider (vendor) would want to charge small amounts on a per-article basis (perhaps varying amounts). The main purpose of the component-based NetPay vendor system was to separate the NetPay EJBs from the particular domain knowledge of the web application, enabling each enterprise bean to be reused in different EJB-based vendor systems via plug-and-play with the existing vendor components. In addition, the E-journal's user interface must be annotated to display E-coin balance, article cost, credit checks and coin debits to the NetPay EJBs.

3 Component-based NetPay Architecture

We have developed a set of Enterprise Java Beans (EJBs) that capture the functionality of the NetPay micro-payment system [3]. These include components such as:

- ArticlePrice. This provides pricing for items or services sold by the vendor.

- EwalletController. This provides sever-side e-coin management, either by managing a customer's e-wallet or providing a CORBA interface to an e-wallet hosted on the customer's own PC.
- RedeemController. Provides nightly redemption functionality to broker.

We have used CORBA to enable the NetPay EJB application servers to access the broker application server to obtain touchstone information to verify the e-coins being spent and to redeem spent e-coins. A dispatcher is used to forward web browser requests to JSP pages. Fig. 1. shows some required interactions of existing E-journal system components (grey) with some NetPay components.

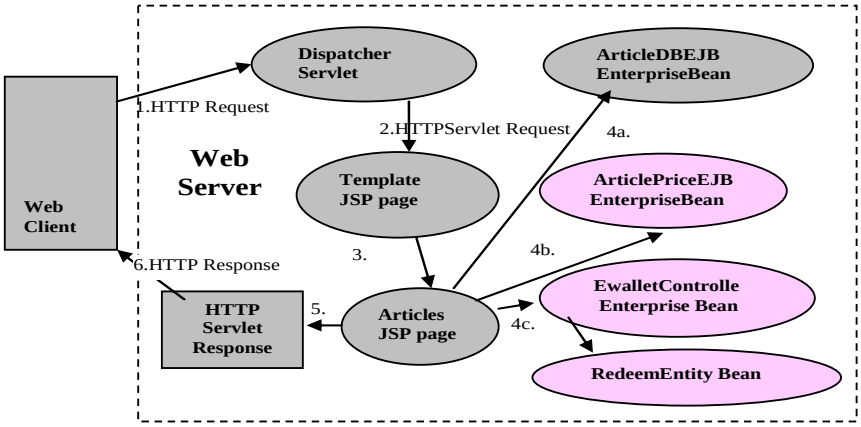

Fig. 1. Web component interaction after modified article.jsp

4 NetPay Integration with E-journal Webpages

In order to add our NetPay micro-payment facility to the E-journal, or to other 3rd party J2EE-based applications, we need to be able to add our EJBs to their J2EE server and to detect when pages are being accessed by customers that need to be paid for. We also need to ensure that if the customer attempting to access does not have enough e-coins they are directed to the NetPay broker site to buy some more. There are three main ways to integrate the NetPay user interface facilities: (1) modify the existing system web pages to incorporate NetPay information; (2) generate web pages that display the existing system pages in frames and make appropriate interactions with NetPay EJB components; and (3) generate proxy web pages that interact with NetPay session beans and redirect access to the original web pages.

4.1 Modifying the Existing System Web Pages

In this approach the articles.jsp is modified to retrieve price data from ArticlePriceEJB enterprise bean for displaying article price information or retrieve e-wallet data (for server-side NetPay) from e-wallet enterprise bean for displaying e-wallet information. Fig. 1 depicts the interaction between these Web components. A HTTP request (1) is delivered to the dispatcher component which processes and then forwards the HTTPServlet request (2) to the template.jsp. The template.jsp generates the response (3) by including the responses from Articles JSP page. Articles JSP page retrieves article contents from the article enterprise bean (4a) and article price from the article price enterprise bean (4b), and e-wallet data from e-wallet enterprise bean (4c). Articles JSP page transmits responses (5 and 6) to the client for presentation.

The content.jsp is modified to make payment from e-wallet enterprise bean in order to debit e-coins paying for article content and Login.jsp is implemented (for server-side NetPay) to retrieve e-wallet data from e-wallet enterprise bean. Fig. 2 depicts the interaction between these components.

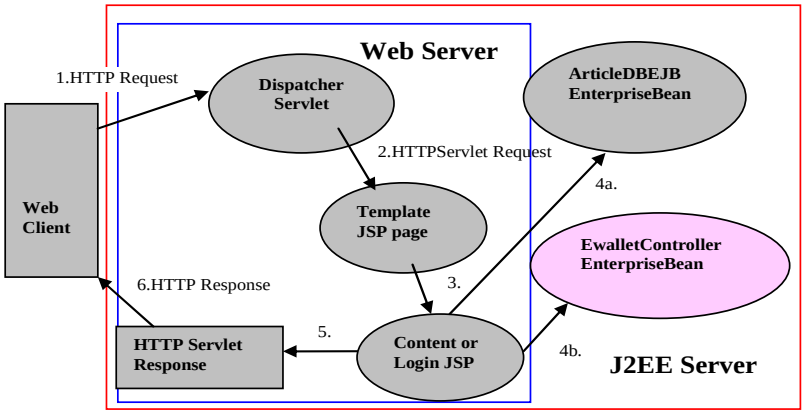

Fig. 2. Web component interaction after modified and implement JSP pages

A HTTP request (1) is delivered to the dispatcher component which processes and then forwards the HTTPServlet request (2) to the template.jsp. The template.jsp generates the response (3) by including the responses from Content or Login JSP page. Content JSP page retrieves article contents from the article enterprise bean (4a) and Login JSP page retrieves e-coin ID and password from the e-wallet enterprise bean (4b). Articles or Content JSP page transmits responses (5 and 6) to the client.

This approach requires updates to the existing system web page implementations. For example, in the journal example system, article.jsp needs to be modified to interact with the price and the e-wallet enterprise beans for displaying the costs of articles and e-wallet balance. content.jsp was modified to debit e-coin from the e-wallet by interacting with the e-wallet enterprise bean before displaying an article content. This can be done easily and possibly by code injection into the existing JSPs by a tool.

4.2 Generating NetPay JSP Pages

In this approach NetPay JSP pages are generated to interface to existing E-journal pages. An HTTP request (1) is delivered to a NetPay JSP page which displays article.jsp in frames, after retrieving article price data from the article price enterprise bean (3) and e-wallet data from the e-wallet enterprise bean (4). The article.jsp retrieves articles' title and author data from the article enterprise bean (2). NetPay JSP pages display the articles and e-wallet information to the client (5). Fig. 3 depicts the interactions between these components. NetPay JSP pages also display content.jsp in frames and interact with the e-wallet enterprise bean in order to debit e-coins.

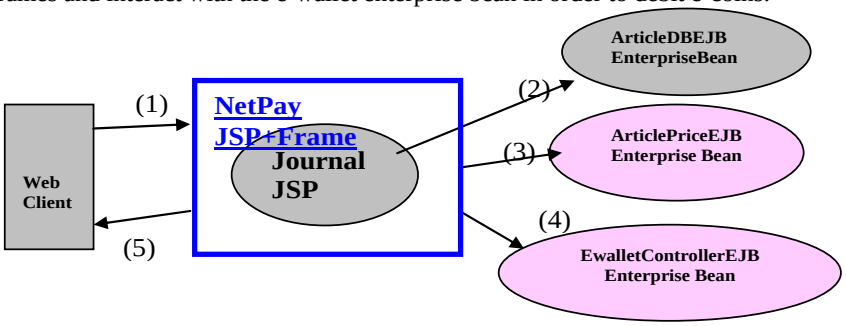

Fig. 3. Generating NetPay JSP pages.

This approach has the advantage that no code changes to the original JSPs are needed and the NetPay pricing information can be displayed in a separate frame to existing information. However, this separation of pricing of items from item descriptions may not be ideal when large numbers of items are displayed together e.g. for large search results, table of contents, news headlines etc.

4.3. Generating NetPay proxy JSP Pages

In this third approach NetPay JSP pages are again generated. These however act as "proxies" to the original web-based system's JSP pages. An HTTP request (1) is delivered to the NetPay proxy pages which obtain article price data from the article price enterprise bean (2) and e-wallet data from the e-wallet enterprise bean (3) for displaying costs of the articles and e-wallet information. NetPay proxy JSP pages then redirect to article.jsp accessing the journal home page (4). When a customer wants to read an article content, NetPay proxy JSP pages interact with the e-wallet enterprise bean to debit e-coins from customer's e-wallet (3) and then redirect to content.jsp which retrieve article content from article enterprise bean (5). Finally NetPay proxy JSP pages display the article content to the client (6). Fig. 4 illustrates the interaction between these components.

This approach hides the NetPay functionality from the user without requiring code changes. However, displaying article price information, information about e-coins left in wallet/spent and so on has to be done by the proxy before forwarding to the original pages. For sites with complex multi-JSP page interactions, this can be intrusive.

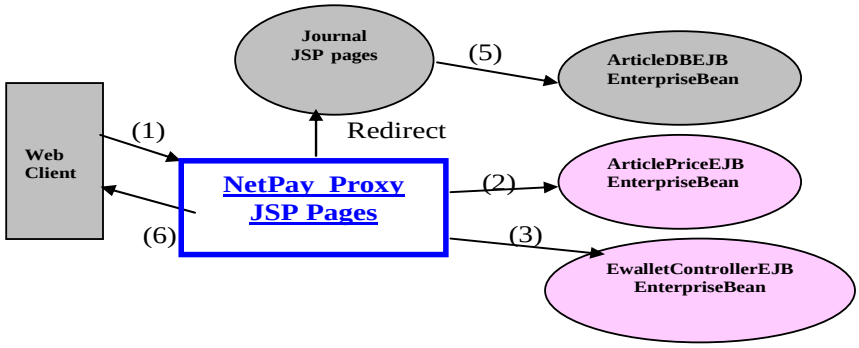

Fig. 4. Generating NetPay proxy JSP pages

5 Summary

We have built NetPay vendor Enterprise JavaBeans to provide plug-in vendor micropayment support components and plugged in EJBs into the E-journal's existing application server and developed three techniques to have the E-journal's JSPs to make appropriate function calls to the NetPay EJBs. These allow for minimal code impact to the existing system's infrastructure. The NetPay vendor system components have been designed, implemented, plugged into the E-journal example system, and successfully deployed to a J2EE server running the E-journal web site.

References

1. Dai, X. and Lo, B.: NetPay – An Efficient Protocol for Micropayments on the WWW. Fifth Australian World Wide Web Conference, Australia, 1999 .
2. Dai, X., Grundy, J.: Architecture of a Micro-Payment System for Thin-Client Web Applications. In Proceedings of the 2002 International Conference on Internet Computing, Las Vegas, CSREA Press, June 24-27, 444-450.
3. Dai X. and Grundy J.: Architecture for a Component-based, Plug-in Micro-payment System, In Proceedings of the Fifth Asia Pacific Web Conference, LNCS 2642, Springer, April 2003, pp. 251-262.
4. Manasse, M.: The Millicent Protocols for Electronic Commerce. First USENIX Workshop on Electronic Commerce. New York, 1995.
5. Rivest, R. and Shamir, A.: PayWord and MicroMint: Two Simple Micropayment Schemes. Proceedings of 1996 International Workshop on Security Protocols, LNCS 1189. Springer, 1997, 69—87.
6. Yen, S-M.: PayFair: a prepaid internet ensuring customer fairness micropayment scheme. IEE Procs-E Computers & Digital Techniques, vol.148, no.6, Nov. 2001, pp.207-13.