An Implementation of Electronic Shopping Cart on the Web System using Component-Object Technology

Satoru UEHARA[†‡]

[†] NTT Data Corporation, Japan
s-uehara@ics.es.osaka-u.ac.jp

Osamu MIZUNO[‡], Tohru KIKUNO[‡]
[‡] Graduate School of Engineering Science,
Osaka University, Japan
{**o-mizuno, kikuno**}**@ics.es.osaka-u.ac.jp**

Abstract

We propose a new mechanism for implementing the electronic shopping cart system (shortly, the shopping cart system) on the World Wide Web system (the Web system). The electronic shopping cart system is one of typical client-server systems, and it includes essential tasks to be implemented in the typical Web based client-server system. The most important task is to maintain context data between successive user sessions.

Although several methods, which can be applied to implement the electronic shopping cart on the Web system, have been proposed, any of them can not attain the task of maintaining context data sufficiently. In this paper we analyze the task and point out the following three difficulties (d1) reliability, (d2) safety, and (d3) session management. We then propose a new mechanism (called the Context Data Storing (CDS) mechanism) to solve all of (d1), (d2) and (d3). In the proposed CDS mechanism, the context data for the session management is stored in the main memory of the client computer. As a result, the CDS mechanism can achieve both high reliability and high safety as well as the management capability of user sessions.

Next, we use component object technology to implement the CDS mechanism. Then, we compared the performance of the electronic shopping cart system using the proposed CDS mechanism with the one using the previous methods. The result showed that our proposed mechanism has solved all difficulties (d1), (d2) and (d3) and has attained efficient communications between clients and Web servers.

1 Introduction

Recently, the World Wide Web (the Web) is becoming the mainstream of the base system on which client-server systems are developed. One of the main reasons for the wide-spreading use of the Web is the advantage in the maintenance cost of the software. Actually, the software developed by the conventional application builder (such as Microsoft Visual Basic) is large in size,and the cost needed for distributing a new version (obtained by version up of original one or modification of bugs) to the client customers is also high. However, for the Web system, all programs are on the Web server and the clients have the Web browsers only. Thus every client can use the newest application software, and the cost of distribution becomes almost zero. As a result, the Web-based client-server system is widely adopted in not only the Internet application but also the intra-net applications.

In the development of the Web-based client-server system, there are several problems that did not occur in the conventional client-server systems. The problems should be analyzed to find an effective way for solving them. Then, this paper takes an electronic shopping cart system as a typical example of the client-server system to be developed on the Web. Though the electronic shopping cart system consists of quite simple functions, it contains essential problems for the construction of client-server systems on the Web. A data sharing among several user sessions is one of such problems.

The HTTP protocol is most commonly used for the construction. It is known that the HTTP protocol is 'sessionless', and thus the connection between a client and a server is disconnected for each communication. When a Web browser accesses a Web server and then successively accesses the server again, the information of the previous access never be maintained. Thus, the shared data among user-sessions must be maintained by any other methods. Hereafter, we call such shared data as "context data." We also call the management of several user sessions by keeping the context data as "session management."

In order to realize the session management and implement the shopping cart system, we will propose a new context data storing (CDS) mechanism using the component object technology. Though several methods have been proposed to realize the session management, they are not sufficient from the viewpoint of reliability and safety of the shopping cart system. In the proposed CDS mechanism, the context data for the session management is stored in the main memory of the client computer. Furthermore, it executes updating of the context data without using any user's updating activities. As a result, the proposed CDS mechanism can realize the session management with high reliability and safety.

We then compared the performance of the electronic shopping cart system using the proposed CDS mechanism with the one using the previous method. The result showed that our proposed mechanism has solved all difficulties and has attained efficient communications between clients and Web servers.

2 Electronic Shopping Cart on the Web

In this section, we define the electronic shopping cart system on the Web (shortly, the Web shopping cart system). The Web shopping cart system can be considered a typical client-server application, since it includes many essential features that are possessed by most client-server applications.

The Web shopping cart system consists of the following entities: customers, shopping carts and items for sale. The customers can do any of the following four actions: (1) browse the item list, (2) select items and put them in their cart, (3) remove items from their cart and return them, (4) send their cart to the accounting server.

Thus the basic behavior of the Web shopping cart system is summarized as follows:

In the Web shopping cart system, there exist several shopping sites on the Web. Actually, the shopping site consists of many Web pages that are linked each other. For each item in the shopping site, its price is assigned. Then a customer can use an electronic shopping cart to hold all the items that the customer has selected to buy.

For the shopping, the customers are allowed to take the following three behaviors: (1) browse freely listed items for sale on the Web, (2) put such item in their carts that customers decided to buy, (3) return freely items from the carts, if customers change their mind,

Finally, at the time when customers decide to stop buying any other items and buy all items in their cart, they send the cart to the shopping server and pay an account.

When we implement the Web shopping cart system, we face several difficult problems. Since the HTTP protocol is 'session-less,' it is difficult to maintain the contents in a cart among successive user sessions. For example, during the shopping, the customer browses items in the shopping site. When the customer goes from one Web page to another, the content of the cart must be maintained. However, the HTTP protocol cannot maintain it even if the customer is in the same shopping site.

There exist other important issues to be considered in the Web shopping cart system. The first is the reliability problem. If the Web server crushes during shopping, the content in the cart may disappear. The contents in the cart should be maintained until the crushed server recovers. The second is the security problem. Since the content in the cart may include secret information, it is preferred that the contents are not viewed by other users. Furthermore, in order to prevent illegal action by the customer, we had better establish a rule that the content (for example, the price of each item) never be changed by the customer.

So far, several methods[3] have already been proposed and implemented on the Web browser. However, those methods can only solve a part of above problems (For more detail, we will explain it in Section 3). In order to solve all these problems, we propose a new mechanism.

3 Implementation by Previous Method

3.1 Previous methods

There are several methods that seem to be useful for managing the context data. The most simple method is just using a parametrical document. The next is generally called HTTP cookie[3, 7]. The "HTTP cookie" usually represents the "client-side cookie", which stores the context data on the client computer. On the other hand, there is a different type of cookie called the "server-side cookie," which stores the context data on the Web server.

3.1.1 Parametrical document

In this method, the context data is described as arguments for a program that generates the dynamic HTML document. Thus it is clearly easy for this method to realize the session management in the Web shopping cart system. Additionally, this method can be used in any platform that can access the Web, since it is completely independent from both the Web servers and the Web browsers.

However, since the context data can be seen in the source of an HTML file, the contents of the shopping cart are easily accessed and thus changed by the users. Thus there may exist serious problem with respect to safety.

3.1.2 Client-side cookie

The client-side cookie is most widely used to manage the context data on the Web. In this method, the text files called "cookie files" are stored on the client computer, and the context data are kept in the cookie files. Once a cookie file is generated in the client computer, the Web server can read and write it.

In order to apply this method, the Web browser should have the functions for the cookies within it[1]. Since we can specify arbitrary the period of storing the cookie files, the cookie files are designed to be stored at the time when the Web browser finishes its execution. Using this property, the Web application can be constructed to resume the situation in the previous visit.

However, in the client-side cookie also, there exists serious problem of low safety. Since the cookie files are stored as a plain text file, the contents of a shopping cart are easily modified by the users.

[1] In these days, most of the Web browsers have such functions.

3.1.3 Server-side cookie

The server-side cookie method is similar to the client-side cookie method. The difference is that the server-side cookie stores the context data on the main memory of the Web server rather than the client computer. On the client computer, just a small cookie file is generated to contain the session's ID only.

In order to use this method, both the Web bowser and the Web server must have the cookie functions. In this method, the context data is stored on the Web server, and thus the users cannot refer the context data. Therefore, the server-side cookie has high security of the context data.

3.2 Difficulties of previous methods

We show an implementation of the Web shopping cart system using the client-side cookie method in Figure 1. Figure 1 represents an activity flow, which selects item1, item2 and item3 one by one and put them in the cart, and finally pays an account. Using the client-side cookie, the cookie file is generated by the Web server's program to store the context data for each item. Thus, the communications between the client and the Web server occur frequently. At time t_i (i =1, 2 and 3), item list is browsed, and item i is selected and put in the cart. Finally, at time t_4, the cart is sent to the accounting server. Please note for each selection of new item, a communication is done between server and the client computer via the Web. If we use other methods such as parametrical document and server-side cookie, then the similar situation as shown in Figure 1 happens between server and client.

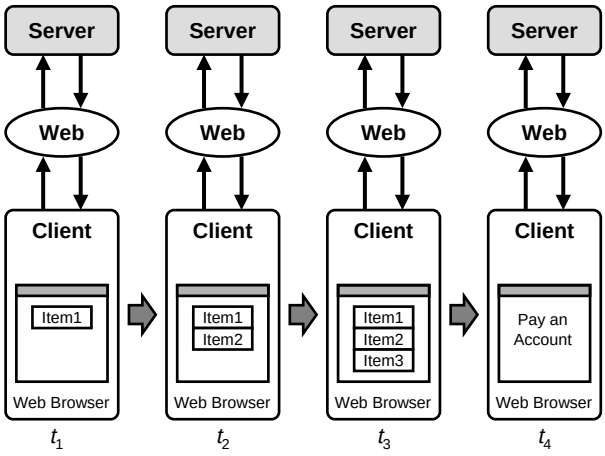

Figure 1. Implementation using client-side cookie

Furthermore, there still exist several other problems. First, consider the server-side cookie method. If the Web server crushes or is down, then all the context data on the server disappear suddenly. In order to avoid such a situation, several Web servers should be introduced. Thus, this method cannot solve the reliability problem of the Web shopping cart system.

Next, consider parametrical document. Then, the context data is shown in the HTML document on the client computer without any encryption. Thus users can easily access the context data. Finally, consider the client-side cookie method. In this method, the context data is stored in plain text files without any encryption on the client computer. If users know the place of the cookie file, then they can refer, modify, or copy the cookie file. Therefore, these methods cannot solve the safety problem.

4 Context Data Storing Mechanism

We propose a new mechanism for implementing the Web shopping cart system. First, we introduce a document-view architecture. Then, based on this new architecture, we define a new mechanism, called "Context Data Storing (CDS) mechanism," to solve those problems mentioned in Section 3.

4.1 Architecture

We introduce a document-view architecture, where the data management part (document) and the graphical user interface part (view) are constructed separately as shown in Figure 2. Then the main functions for the Web shopping cart system can be constructed effectively by assigning the context data and the Web browser to the document and the view, respectively.

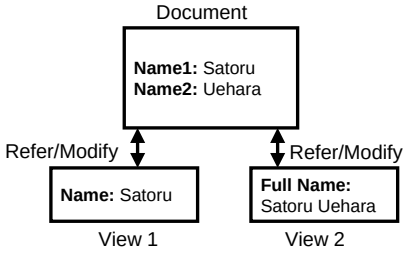

Figure 2. Document-view architecture

Furthermore, users can refer the document in their own ways by defining the views. Thus the usability of an application software is also improved.

Figure 3 shows updating of data in a view. If several views exist in the same system, updating of data from one view must be informed to all other related views. As shown in Figure 3, suppose the data named "Name" on the View-1 is modified. Then the corresponding data "Name1" on the document must be updated and its change must be informed to the View-2.

In order to implement the document-view architecture, we used two kind of communications, based on the

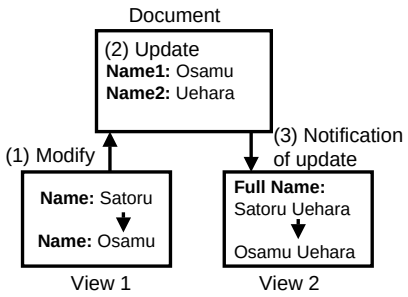

Figure 3. Updating of data in a view

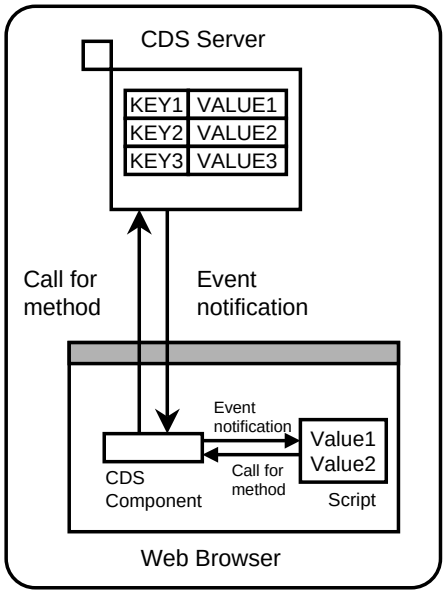

Figure 4. Logical organization of CDS mechanism

COM[10] (which is the component object technology by Microsoft): (1) call for method, and (2) event notification. Figure 4 shows a logical organization of the CDS mechanism, which consists of CDS component and CDS server.

a) CDS component

The CDS component is a software embedded in the Web browser. It works for an interface between the CDS server and the Web browser. Actually, it receives the call for a method from the Web browser, and sends the notification of an event to the Web browser.

The event handling is performed as follows: 1) When the data in the CDS server is updated, the CDS server sends a notification of the event (that is, updating of the data) to the CDS component. 2) On receiving it, the CDS component successively sends the notification to the Web browser. 3) The Web browser updates the corresponding data according to the notified event.

In the proposed implementation, we realize the CDS component as the dynamic link library (DLL).

b) CDS server

The CDS server is a software that manages the context data. It receives the call from the CDS component for a method to access the data, and sends the notification of the event to CDS components.

It is implemented as a different executable process from the Web browser. Thus, even if a Web browser crushes down, the CDS server still works alive and can keep the context data. Additionally, in the proposed implementation, we store the context data on the main memory of the client computer. As a result, the contents of the context data are hardly accessed by the users (by any illegal method).

The CDS component and CDS server are downloaded into a client computer via the Internet. The signatures in the component should be certificated by a trusted organization.

The details of the implementation of the CDS mechanism is shown in Appendix.

4.2 Basic behavior of CDS mechanism

In order to execute the CDS mechanism, the invocation command of the CDS component must be described in a HTML document. Then the CDS component is loaded into the Web browser's process. At the same time, the CDS server is also invoked in a different process (from the Web browser's process) and it starts managing the context data.

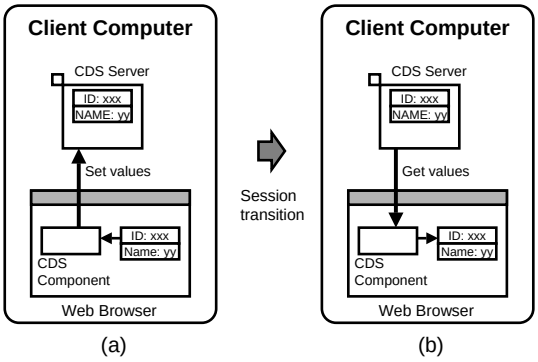

Figure 5. Maintenance during successive sessions

Here, we explain the basic behavior of the CDS mechanism using Figure 5. This is an example to maintain a user's ID and a user's name during successive sessions. First, assume that user inputs ID and name. Then, the CDS component and the CDS server are invoked at the same time. The CDS component calls the method to create the keys, such

as “ID” and “NAME”, and to set the value for each key (for example, ID=xxx and Name=yyy as shown in Figure 5(a)). Next, assume that the current session transits to the next session (as shown in Figure 5(b)). Just after the transition, the context data such as “ID” and “Name” are not maintained on the Web browser. Then, the CDS component automatically calls the method of the CDS server to get these context data.

As explained in the previous example, updating event of the context data is notified to the Web browser automatically by the CDS mechanism. On the other hand, in the conventional Web systems, the data in a view must be updated by a user’s action such as clicking “reload” button. The proposed mechanism can handle such events, and thus can update the view without employing any user’s action.

4.3 Lifetime management

Since the CDS mechanism uses the main memory of the client computer, the lifetime management is important. That is, the CDS component should be on the main memory during the time when the HTML document is displayed on the screen, and thus it should be unloaded as soon as the user session finishes. To do so, we must insert an appropriate `<OBJECT>` tag into the HTML document.

For example, the `<OBJECT>` tag is specified as follows:

```
<OBJECT ID="objCDS"
 CLASSID=
 "CLSID:D85F06A4-53A8-11D3-ABFE-
00A0C9A3C303"
 CODEBASE="CDSComponent.cab\#version=1,0,0,1"
</OBJECT>
```

Here, for the corresponding CDS component, `ID`, `CLASSID` and `CODEBASE` denote the ID to be referred from the scripting language, the unique ID in the client computer, and place and version of the program, respectively. To unload the CDS component, we just remove the `<OBJECT>` tag from the HTML document.

On the other hand, the CDS server is loaded while the CDS component is on the main memory. Only when all of the CDS components in the client computer are unloaded, the CDS server also unloaded. Thus for a lifetime management in the proposed implementation, we used the reference counter based on the COM[10], which indicates the number of references by other objects. If the value of reference counter becomes 0, then the component object is unloaded automatically.

5 New Implementation of Shopping Cart

5.1 Implementation using CDS mechanism

Next, Figure 6 shows an implementation of the Web shopping cart system using the proposed CDS mechanism. Consider the same shopping cart example as the activity flow in subsection 3.2. At the time t_i (i =1, 2 and 3), item list is browsed and item i is selected and put in the cart. Finally, at time t_4, the cart is sent to the accounting server. In this implementation, the contents of the cart are stored in the main memory by the CDS server process. Since the context data can be accessed from the Web browser using the scripting language, the number of accesses to the Web server from the client decreases, as shown in Figure 6.

Now we explain how CDS mechanism works in the implementation. At time t_1, the CDS server is loaded to the main memory and is executed as a different process from the Web browser. Then the CDS server downloads the item list from the Web server. At times t_2 and t_3, the client accesses to the context data via the CDS component. Please note that these accesses need not any more communications with the Web server. At time t_4, the cart holding all selected items are sent to the Web server to pay account.For this action, the client needs a communication with the Web server via the Web.

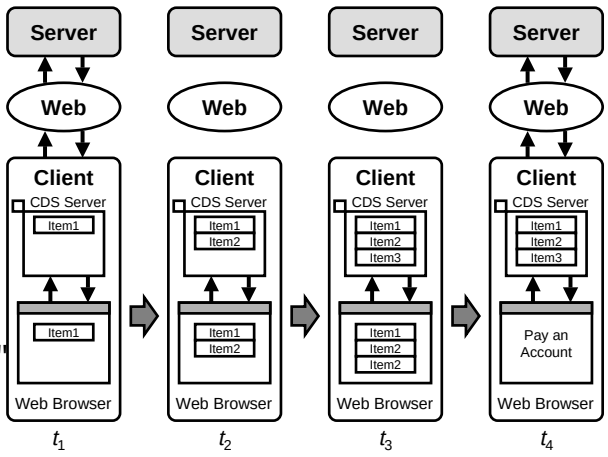

Figure 6. Implementation using CDS mechanism

5.2 Advantages of the CDS mechanism

The example (shown in Figure 6) shows the CDS mechanism can attain the reliability and safety of the Web shopping cart system. Furthermore, the CDS mechanism can mitigate the communication overhead between the client computer and the Web server. As mentioned before, the Web shopping cart system is considered to be a typical client-server application, we can expect that the proposed CDS mechanism is useful for developing the client-server systems on the Web.

6 Conclusion

In this paper, we defined the Web shopping cart system as a typical client-server application on the Web. We then clarified several problems on the implementation of the Web

shopping cart system, which are peculiar to the Web. In order to solve the problems, we proposed a new mechanism that can manage user sessions with high reliability and safety. Then we compared the Web shopping cart system implemented using the proposed mechanism with the one developed by the conventional methods. The result shows that our mechanism can successfully implement the Web shopping cart system.

Future research includes quantitative analyses for the advantage of the CDS mechanism by applying it to much more practical developments.

Acknowledgment

Authors would like to thank Mr. Chikara Nakano for his cooperation

References

[1] Apache Software Foundation, Apache HTTP Server Project, *http://www.apache.org/httpd.html*.

[2] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk and T. Berners-Lee, "Hypertext Transfer Protocol - HTTP/1.1," RFC 2068, Network Working Group, 1997.

[3] D. Kristol and L. Montuli, "HTTP State Management Mechanism," RFC 2109, Network Working Group, 1997.

[4] Microsoft, Internet information services features, *http://www.microsoft.com/windows2000/guide/server/features/web.asp*

[5] Microsoft, Active Template Library (ATL) Reference, *http://msdn.microsoft.com/library/devprods/vs6/visualc/vcmfc/atl.htm*

[6] Netscape Inc., Netscape Enterprise Server, *http://www.netscape.com*

[7] Netscape Inc., HTTP Cookies, *http://home.netscape.com/newsref/std/cookie_spec.html*

[8] H. Raggett, A. Le Hors and I. Jacobs, "HTML 4.0 Specification," W3C Working Draft, 1997.

[9] D. Robinson, The WWW Common Gateway Interface Version 1.1, Internet draft, 1995.

[10] D. Rogerson, *Inside COM*, Microsoft Press.

[11] N. Yeager, R. McGrath, *Web Server Technology, The advanced guide for World Wide Web Information Providers*, Morgan Kaufmann, 1996.

Appendix A

Implementation of the CDS mechanism

In this appendix, we will describe the implementation of the CDS mechanism.

The CDS mechanism runs on client computers, and it is handled by a script language on the Internet Explorer. Such scripts are described by server-side programs. The developers can develop server-side programs with various languages — ASP, JSP, PHP, Java and so on.

There are the following 7 major functions to be manipulated by the server-side programs: a) InitializeServer method, b) Remove method, c) RemoveAll method, d) Item property, e) Count property, f) Name property and g) OnDataChanged event. By applying the methods a), b) and c), the context data is newly created or removed. The content of the context data can be obtained by the properties d), e) and f). Finally, updating of the context data is notified by the event g).

a) InitializeServer method

An `InitializeServer` method initializes the CDS server. Its specification is described by the IDL (Interface Definition Language)[10] as follows:

```
[id(0), helpstring("Initialize")]
HRESULT InitializeServer
  ([in] BSTR bstrID);
```

This method takes a string type as an argument. In the CDS server, the context data is stored in the form of a table having unique keys and their values. We call such table as "context data container."

The context data container is prepared for each invocation of the method with different argument. That is, an argument for the `InitializeServer` method is an identifier of the context data container. If the `InitializeServer` is invoked with an existing identifier, it points to the existing context data container. This feature makes it possible to share a context data container among several Web browsers.

Figure 7 shows an application of this method to share a context data container. In this example, the Web browsers 1 and 1' share a context data container with the identifier "Browser1." The Web browsers 2 and 2' also share another context data container with the identifier "Browser2."

b) Remove method

This method removes the context data that is no longer used. It takes a string type as an argument, which represents the key of the context data. The specification of the method is described as follows:

```
[id(2), helpstring("Remove item")]
HRESULT Remove([in] VARI-
ANT varName);
```

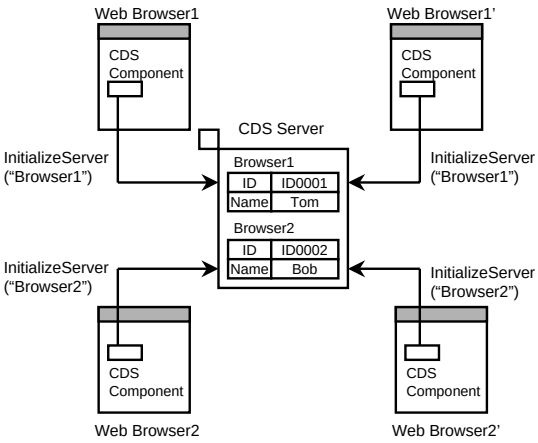

Figure 7. InitializeServer method

For example, in order to remove the context data with the key "Name," the following script is described:

```
<SCRIPT LANGUAGE="VBScript">
  objCDS.Remove("Name")
</SCRIPT>
```

c) RemoveAll method

This method removes all context data in a context data container. Its specification is described as follows:

```
[id(3), helpstring("Remove all")]
HRESULT RemoveAll();
```

The following script can remove all data:

```
<SCRIPT LANGUAGE="VBScript">
  objCDS.RemoveAll
</SCRIPT>
```

d) Item property

This property is for setting or getting the context data stored in the CDS server.

```
[propget, id(1), help-
string("Get item")]
HRESULT Item
  ([in] VARIANT varName,
   [out, retval] VARIANT *pvarVal);

[propput, id(1), help-
string("Set item")]
HRESULT Item
  ([in] VARIANT varName, [in] VARI-
ANT newVal);
```

The former specification is for getting the data, and the latter is for setting it. For example, two context data "ID" and "Name" are stored in the CDS server by the following script:

```
<SCRIPT LANGUAGE="VBScript">
  objCDS.Item("ID") = "ID0001"
  objCDS.Item("Name") = "Tom"
</SCRIPT>
```

On the other hand, to get the values of the context data from the CDS server, the following script must be written:

```
<SCRIPT LANGUAGE="VBScript">
  strID = objCDS.Item("ID")
  steName = objCDS.Item("Name")
</SCRIPT>
```

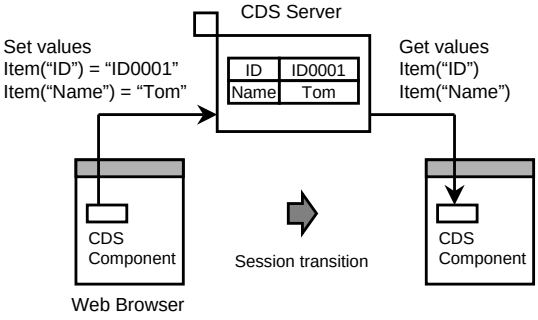

Figure 8. Item property

Figure 8 shows an application of `Item` property. In this example, two context data "ID" and "Name" are stored into the CDS server using `Item` property. Next, after the session transition, the context data are obtained using `Item` property.

e) Count property

This property stores the number of context data. Its specification is described as follows:

```
[propget, id(4), help-
string("Count of data")]
HRESULT Count([out, ret-
val] long *plVal);
```

The following script can acquire the number of context data in a context data container:

```
<SCRIPT LANGUAGE="VBScript">
  Dim nCount
  nCount = objCDS.Count
</SCRIPT>
```

f) Name property

This property gets the name of context data by the index. Its specification is described as follows:

```
[propget, id(5), help-
string("Name of data")]
HRESULT Name([in] long lIndex,
[out, retval] BSTR *pbstrVal);
```

The following script can acquire the name of context data:

```
<SCRIPT LANGUAGE="VBScript">
  Dim strName
  strName = objCDS.Name(1)
</SCRIPT>
```

g) OnDataChanged event

This event occurs when the context data in the CDS server is changed. Its specification is described as follows:

```
[id(6), helpstring("OnDataChanged event")]
HRESULT OnDataChanged();
```

When the context data is changed, `OnDataChanged` event is notified from the CDS component to the CDS server. This event is only notified to CDS components that initialized the context container in which the context data was changed.

Figure 9 shows an application of OnDataChanged event. When the context data in the container "browser1" is changed, `OnDataChanged` event is notified to the CDS components in the Web browsers 1, 1' and 1".

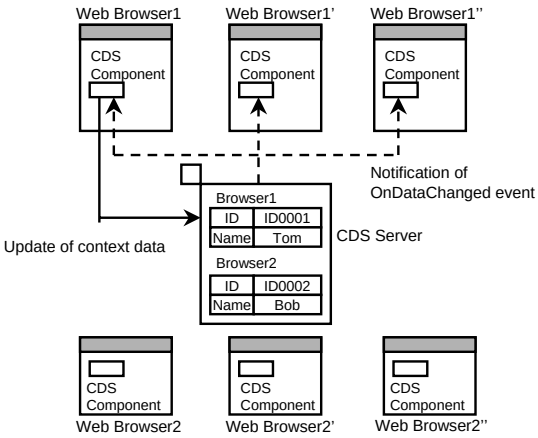

Figure 9. OnDataChanged event

In order to use the `OnDataChanged` event, the following script must be described in an HTML document:

```
<SCRIPT LANGUAGE="VBScript">
Sub objCDS_OnDataChanged()
  ' Describe the updating rou-
tine here
End Sub
</SCRIPT>
```