Managing State in 4D Web Systems

The materials in this document are from the *4D Internet Integration* training course I am presenting internationally during 2000. Given the enormous interest expressed in this course from people who cannot attend, I am providing this sample chapter for your use. The cross-referenced chapters and code samples are included in the full course. I am interested in hearing your reactions to this material. Please email me your comments and questions at:

dpadams@island-data.com

CHAPTER 27

Managing State

Introduction

The Web was not designed with database applications in mind. The Web was designed with the assumption that users would follow links to servers all over the world, not spend a lot of time interacting with a single server. This is inconvenient when you move database applications to the Web. Imagine a typical database task:

The user performs a search.

The program shows the user the list of records matching their search.

The user selects a record to look at in detail.

The program shows the user the record he or she asked to request.

The user edits the record and asks the program to save it.

The program saves the user's changes.

This is easy in 4D because the program always knows who the users are and which records they are working with. With a traditional Web server—or a traditional database integrated with a Web server—no information is retained between actions. After the search is done, the search results are not saved anywhere on the server. When the user asks to edit a record, the process starts over. This chapter looks at state management and various strategies for implementing it.

Elements of State Management

Introduction

A state management system includes at least three elements:

- Data
- Name
- Persistence

After explaining each of these elements, the chapter looks at different ways to implement each one.

Data

Data is the heart of a state management system. Because you are working in a database environment, you have a huge advantage over Web developers working with traditional Web tools. You can store whatever data you need to implement your system in 4D. The following are examples of the kind of information you can store:

- ✓ The name and access rights of the current user.
- ✓ A description of the user's most recent search.
- ✓ A set or named selection with the results of a user's search.
- ✓ Historical information about the user's past behavior at your site.
- ✓ The user's preferred language.
- ✓ The user's email address and other contact information.

Name

The Web database system and the Web browser need to coordinate access to the stored state data. This is done with a name. You can build a name for this purpose in many ways. The only requirement is that you can use the name to find the correct state information. If you use an ecommerce Web site you will see URLs that embed information that allows the Web database system to coordinate the browsers activity with the state information stored on the server:

http://shop.barnesandnoble.com/Shop/cart.asp?userid=48SBFU8VOB&mscssid=LBQBAELFURSH2PQM00JP42CB5R889C2E&sref=&xt=Y&direct_to_summary=false

4D's contextual Web server includes a context and subcontext ID as a name:

http://www.foo.com/%23%231596484872.2/4Ddisprec/0

Persistence

By default Web servers do not retain any information about past requests. The Web server receives the request, responds to the request, and the forgets about the request. The request information has zero persistence. The purpose of building a state management system into a Web database application is to add persistence. The state system maintains information about users' requests to tie various actions together in one session. Examples of this feature include shopping cart systems that remember a users' account information and purchases, or a search system that lets the user ask for different pages of search results without entering the same search conditions again. When you design a state management system, you must decide what will persist and for how long.

Data

Introduction

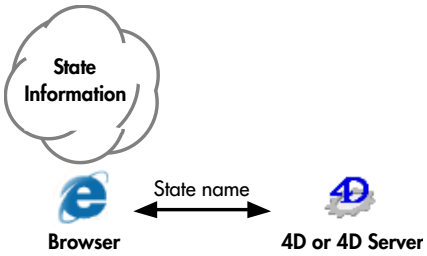
The most interesting Web applications keep track of what the user does over time. The user can add items to a shopping basket, review an order, or go back to an earlier search. Since this information is not stored by traditional Web servers, it has to be stored somewhere else. State information can be stored on the client's machine in cookies or hidden fields, or on the server machine in your database.

Storing State Information in Cookies

Cookies are similar to records or variables. A cookie has a name and a value. Here are two examples:

```
CookieName=CookieValue
AV_USERKEY=SAV0000003896785FD834FB0AKTZYXHM
```

A Web site or a JavaScript can create and read cookies on the client's machine. Cookies can store state information like the users' account numbers or the items in their shopping basket. If you build a pure cookie-based state management system, the state information is all maintained on the client machine.



Storing the state information on the client machine in cookies.

Cookies are a natural approach for developers who do not have access to a database application on the server machine to store data. If you are building a 4D based Web application, you have a database to store information in. Put another way, cookies are a solution to a problem you do not have: where to store data without control of the server. If you have control of the server machine, then cookies are not that appealing because of their disadvantages:

- Users can disable or reject cookies in their browser.
- Some browsers do not support cookies.
- The same cookie may or may not work with different browsers. "Different browsers" means any two browsers that differ in any way: manufacturer, platform, and version. Identically named and numbered browsers on different platforms, for example, can handle cookies quite differently.
- Netscape's original cookie definition was not introduced as a standard. Subsequently, a modified cookie standard has been introduced. There are now at least two different kinds of cookies in use on the Web. The new cookie specification is better than Netscape's original proposal, but having two formats at work only increases the number of compatibility problems you are likely to encounter.
- Cookies are associated with a *machine*, not with a user. This can be a security risk if the machine is unprotected, and an inconvenience if the user does not always use the same machine.

For more information about cookies see the "Cookies" discussion in the "Parsing Incoming HTTP" chapter, starting on page 169.

The WebConnection_ParseCookies *and* Cookie_BuildNetscapeCookie *methods in the* 4D Internet Integration Example *database read and write cookies.*

Storing State Information in Hidden Fields

HTML includes a kind of field called "hidden. Hidden field data is sent to the browser, but not displayed. If the user views the source of the page, they see the hidden field listed in the HTML, like this:

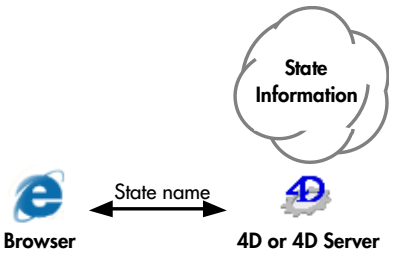
```
<input
type=hidden
name=AV_USERKEY
value=SAV0000003896785FD834FB0AKTZYXHM
>
```

You do not need to put the `input` *attributes on separate lines. HTML is flexible about where you insert white space in the HTML source. You may put the* `input` *definition on a single line if you find that more readable or easier to work with.*

The advantages of hidden fields are that they work in any browser that supports forms, and are easy to define in your HTML source documents. The disadvantages of hidden fields is that they only work when the user submits a form. They do not, for example, work when the user presses a link.

Storing State Information in a Database

State information can also be stored in a database on the server machine. For example, you can save the search results in a set in case the user wants to look at a different part of the search results later.



Storing the state information on the server machine.

You can store the state information you are tracking in any object and location that 4D has access to:

- Variables
- Arrays
- Records
- Lists
- BLOBs
- Documents
- A remote ODBC data source
- A remote 4D Server database reached through 4D Open

You can combine these data storage types and locations in any combination that serves your needs. The advantages of storing information on the server are that you have complete control and can reduce the amount of information sent back and forth between the browser and the server. The disadvantage of storing state information on the server is that it uses up server resources, like RAM and hard disk space.

ObjectTools

One of the easiest and most flexible approaches to storing state data is to use an ObjecTools object. ObjectTools is a plug-in product from Automated Solutions Group:

http://www.asgsoft.com/

ObjectTools lets you store anything you want in an object. You can store pointers, pictures, BLOBs, arrays, strings, dates, and all of 4D's other data types. The advantages of using ObjecTools to store your state management data are:

- You read and write information to and from the object by name. You do not need to use distinct variables or calculate offsets within a BLOB. ObjecTools is easy.
- You can store ObjecTools objects in records for later reuse or to pass between processes or machines.
- You can store different information in different objects. Objects are more flexible than records in 4D tables.

This example creates an ObjecTools object and then adds a date and a string to the object:

```
 ` Create object to store state information.
$object_l:=OT New

 ` Store the current date.
OT PutDate ($object_l;"Last updated data";Current date(*))

OT PutString ($object_l;"Search conditions";$userSearchConditions_s)
```

The value of $object_l identifies the object and lets you read its contents. Here is how you read the data and the string:

```
$lastUpdated_d:=OT GetDate ($object_l;"Last updated data")
$searchCondtions_s:=OT GetString ($object_l;"Search conditions";$userSearchConditions_s)
```

Working with arrays, BLOBs, pictures, and 4D's other data types is no harder. The key to making this work in your Web database application is passing the ObjectTools reference back and forth between 4D and the browser. The next section discusses strategies for managing the exchange of names.

If you use ObjecTools in your state management system, note that the object references are longints. You need to convert them to strings to send to the browser and convert them to longints when received from the browser.

Name

Introduction

Storing state information in cookies or database data on the server machine does not change the nature of the Web. The Web is still stateless. It does not do much good to store the state information if you cannot associate it with the correct user later. There needs to be some way for the browser and Web system to know which state information to use with new requests. There are three common strategies for addressing this requirement:

- Cookies
- Smart URLs
- Hidden fields

This section explains each of these strategies.

Cookies

You can use a cookie to invisibly pass a session or user ID to the server machine. If, for example, you are using ObjecTools, you might use a cookie that looks like this:

```
StateTrackingObjectID=65684640
```

When you send requests out of 4D, you add this cookie, and when requests come back from the browser, you read the cookie.

See "Storing State Information in Cookies" on page 186 of this chapter for comments on the disadvantages of cookies.

The WebConnection_ParseCookies and Cookie_BuildNetscapeCookie methods in the 4D Internet Integration Example database read and write cookies.

Smart URLS

When you visit dynamic Web sites, you will often notice that the URLs include code numbers and other information:

http://www.amazon.com/exec/obidos/subst/home/home.html/002-3116371-0877045

The browser sends the URL to the Web site; the database on the Web site extracts the code from the incoming URL and looks up the state data that this code number refers to. There are two common strategies for embedding a state data name in a URL: embedding a list of arguments and embedding a reference. This first example stores a list of arguments:

http://shop.barnesandnoble.com/Shop/cart.asp?userid=48SBFU8VOB&mscssid=LBQBAELFURSH2PQM00JP42CB5R889C2E&srefer=&xt=Y&direct_to_summary=false

Here is how this might look using an ObjecTools ID:

http://www.foo.com/Products?StateTrackingObjectID=65684640

This approach is convenient when you have a number of arguments—like search strings and preferences—to include. Embedding a reference can be simpler:

http://www.foo.com/Products?65684640

The advantage of smart URLs is that they work with all browsers and are easy to build. The limit is that a single URL is only specified to store up to 1K (1024 characters) of text. Some versions of some browsers do not support the full 1K of text.

See the"Parsing Incoming HTTP" chapter, which starts on page 167, for more information on parsing incoming URS and references to the 4D Internet Integration Example *methods that parse URLs.*

HIdden Fields

Another way to pass the name between the server machine and the browser is with a hidden field:

```
<input
type=hidden
name=StateTrackingObjectID
value=65684640
>
```

The advantages and disadvantages of this approach are the same as when using hidden fields to the state data directly: this strategy works on a wide variety of browsers, but only when the user submits a form.

Persistence

Introduction

The purpose of a state management system is to make state information persist. When you implement state management, you need to address several questions:

- How long does the information need to persist?
- Is this information going to be shared by multiple processes?
- Is this information going to be shared by multiple machines?
- How do you want to handle requests for information that has expired?

Length of Time

How long does your state information need to persist? When you start a 4D process, the variables, named selections, sets, and other process objects persist as long as the process is alive. Once the process stops, the objects are lost with it. When you save a 4D record, it persists until it is deleted even if it is not used for years. When you design your state management system, you are likely to find that different pieces of information need to persist for different lengths of time. This indicates that you are managing different kinds of data. For example, if your system works with customers, you will want to store the customers' names and addresses permanently but may only want to store their recent search history temporarily. The length of time information needs to persist drives where you store the data. Permanent data must be stored in records or documents; temporary data may be stored in variables and arrays.

Multiple Processes

Is your Web handling system going to handle each request from a specific session in a specific 4D process, or will you have multiple processes handling requests in whatever order they are received? The 4D contextual Web server dedicates a specific process to each Web session. This allows the contextual Web server to store state information in 4D process objects, like the current selection, and variables you create. Most Web servers maintain a pool of Web handling processes that handle requests as they arrive. During a Web session a browser may end up working with ten different processes on the server machine.

You design dedicated process and multiple-process solutions differently. Multiple-process solutions are ultimately more flexible but change how you must store the state data. The locations where you can store data that different processes can reach are records, interprocess named selections, interprocess sets, interprocess variables, and interprocess arrays. You may copy or reuse some of this data into process data while handling a request, but you need to keep the data in an interprocess location between requests.

Multiple Machines

Is your Web handling system running on one machine or multiple machines? The built-in 4D Web server always runs on one machine. You may have multiple machines if you use multiple copies of 4D to connect with a server using 4D Open, or if you implement a custom Web server using ITK or Net Tools. If you are designing a multiple machine system, you must decide how to handle Web sessions. You can either configure the system to route requests to the machine that handled the sessions's original request or handle the requests on whatever machine the request reaches. If you want to route the requests to the machine that handled the original request, the easiest technique is to embed the machine name into the URLs you send the browser:

```
http://www3.foo.com/Products?65684640
```

If you want to handle requests on whatever machine is available, you need to store your state information in a location that all of the machines can reach. Database records are the obvious location. If you are using ObjecTools, or a similar solution, you can store the entire collection of state information in one record for transfer amongst machines.

Expired Data

What do you want to do if the user bookmarks data you no longer have? For example, what if the user bookmarks an old search results page? If your system no longer has the information you need to handle the situation. You can send the user an error message or attempt to rebuild the request. If, for example, you embed the original search conditions in the URL, you can perform the search again:

http://www.foo.com/Products?StateTrackingObjectID=65684640&Action=Search&Table=Products&SKU=BAN

State Tracking Tools

In this chapter I have explained the elements involved in designing a state tracking system and proposed approaches to each aspect of the design. An alternative to building your own system is to purchase or use a pre-built system. The 4D contextual Web server is one example of a Web state management system that you can use immediately without coding your own solution.