

## **UNDERSTANDING TECHNIQUES OF SESSION MANAGEMENT FOR WEB APPLICATION COURSES**

*Xue Bai, Virginia State University, [xbai@vsu.edu](mailto:xbai@vsu.edu)  
Qidong Cao, Winthrop University, [caoq@winthrop.edu](mailto:caoq@winthrop.edu)  
Steve Davis, Clemson University, [davis@clemson.edu](mailto:davis@clemson.edu)*

### **ABSTRACT**

*Sharing information as a user travels from page to page is important for designing web applications. The stateless nature of HTTP requires a method to uniquely track a visitor to a web site. This paper discusses several session management techniques for sharing state information over the stateless http protocol. Also it describes how students acquire a better understanding of these session management techniques, including their differences. Students were juniors and seniors majoring in computer information systems in a web application development course at a state university in the United States.*

**Keywords:** HTTP, web applications, web design

### **INTRODUCTION**

#### **Session Management and the HTTP Protocol Limitations**

All interactive commercial web sites require some user-related information be maintained through successive page requests. Unfortunately, the HTTP protocol is stateless and there is no built-in mechanism to share user information through those requests. Maintenance of a state across multiple requests is not supported within the protocol itself. Thus, each time a client requests for a web page, it opens a separate connection to the web server. There is no connection or information sharing among those requests. After receiving a request, the server processes the request and sends response to the client, and then the server forgets all about the request. The next time it receives a request from the client, the request is treated just the same as if it were the first from that client. What if you want the server to hold some information that is particular to a client across multiple requests? For example, when a client is running an on-line shopping system, each request could add an item to the client's ongoing order. The order must maintain its state so it knows what's already in it. Similarly, when a client decides to proceed to checkout, how can the server determine which of the previously created orders is for this client? Because the HTTP protocol is stateless, the state information must be managed another way.

#### **What is Session Management?**

Session management is the process of maintaining state information across multiple requests. The idea is that all of a user's requests for pages from a specific Web server during a given period of time are actually part of the same interactive session and some user-related information is maintained and shared during the same interactive session. For example, to access your email account, you are required to login with your account information, typically, your user name and password. Upon logging in successfully, your account information should be maintained through session management. In successive requests, the session management makes this information available on a user specific basis.

#### **Session Management Techniques**

There are several solutions to the session management problem: 1. query string; 2. hidden fields to carry information during successive requests; 3. cookies to store and retrieve information in successive requests; 4. session object; 5. URL rewriting.

---

## **Teaching Issues**

It has been a challenge to teach students these session management techniques. Based on our teaching experience, students were confused about the difference of these techniques and how to make a selection to implement session management when developing web applications. To help students understand the difference among these techniques and implement the session management when developing web applications, we explained how these solutions store and share information, how the information is carried over through successive requests, and factors that need to be considered when implementing the session management for web applications.

## **STATE MANAGEMENT**

In this section, we examine the techniques that are commonly used to maintain state information across page requests. Rationale and structure of each option are explained, and weaknesses are pointed out. We also provide examples to illustrate how those techniques are implemented in JavaServer Pages (JSP).

### **Query Strings**

A query string is information that is appended to the end of a page URL. A typical query string might look like the following example:

`http://localhost:8080/querystring.html?school=business&major=CIS`

In the URL path above, the query string starts with a question mark (?) and includes two attribute/value pairs, one called "school" and the other called "major." Query strings provide a simple but limited way to maintain state information. For example, they are an easy way to pass information from one page to another, such as passing school and major information from one page to another page where it will be processed. However, some browsers and client devices impose a limit on the length of the URL. For example, maximum URL length is 2,083 characters in Internet Explorer. Information that is passed in a query string can be tampered with by a malicious user. Any person using the same computer will be able to review the browser history file and follow the same URL. . Additionally, a user can bookmark the URL or send the URL to other users, thereby passing that information along with it. Therefore, a programmer should not rely on query strings to convey important or sensitive data.

Many Web sites have accounts for users with personal information or the ability to control various things on the site. For this the site employs a username and password combination to verify that the client who tries to login is truly the real user. After user logs in, the site needs to keep track of the user so that he/she will not need to go through the login process for each page. For this purpose, the site must keep authentication information through one of the session management techniques discussed in this paper. This authentication information will be used on each page request to access the user's account information, such as emails, academic records, and so on. To illustrate how each session management technique handles the state information, an authentication page, typically, a login web page, is designed to show how the authentication information is stored and for later retrieval in successive web page access. Figure 1 shows a typical login page, which accepts user ID and password. There is a list of session management techniques that could be used for storing user authentication data.

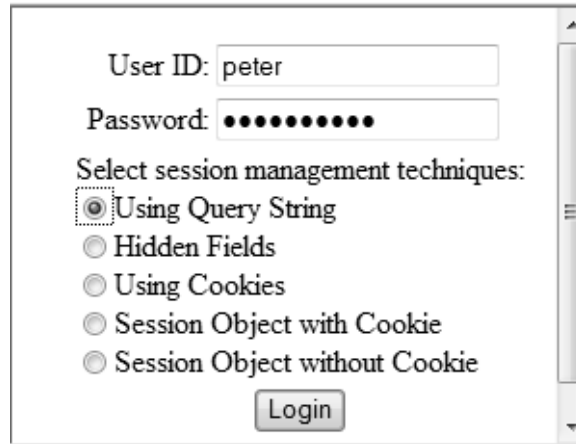


Figure 1. Authentication Login Page

If the query string technique is used to carry authentication data across multiple web page requests, the data will be retrieved in each page access and the same data must be attached to the end of each URL in order to maintain the state information as shown in the following code segment:

```
<a href="retrieve_querystring.jsp?userID=peter&pwd=mypassword">
```

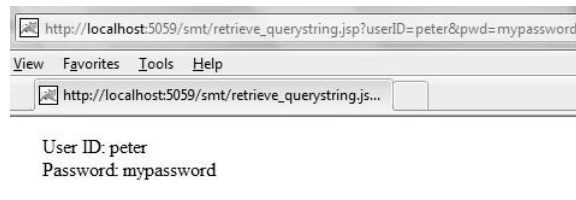


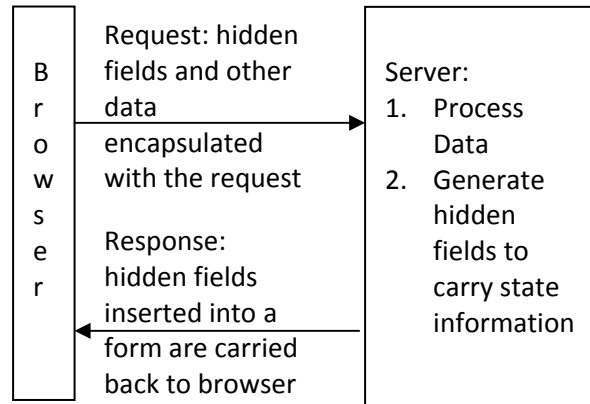
Figure 2. Carry State Information with Query String

As indicated in Figure 2, user id and password are attached to the end of URL. Therefore, any person will be able to review the browser history file and follow the same URL to access the user's account information.

### Hidden Fields

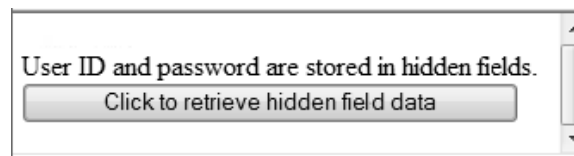
One of the most basic ways of preserving state in web pages is to place data in hidden fields. Hidden fields are simply a type of text field, but they are not displayed on the web browser. A hidden field stores information in its value property. In order to carry data in the successive requests, all hidden fields must be dynamically generated in an html form that is going to be submitted to the server for processing. When a page is submitted to the server, the content of a hidden field is sent in the HTTP Form collection along with the values of other controls. A hidden field acts as a repository for any page-specific information that you would like to store directly in the page. After data being processed and inserted into a form, the data carried over by hidden fields is sent to the client browser. The server will not store any information about those hidden fields.

In order for the hidden field values to be available during page processing, the page must be submitted using an HTTP post method and the page request carrying the hidden fields cannot be interrupted by closing the browser or visiting other page that does not carry hidden fields. Otherwise, all hidden fields may be lost. Hidden fields cannot be utilized if a page is processed in response to a link or HTTP GET method. Figure 3 illustrates how hidden fields are used to carry data over multiple requests.



**Figure 3.** Maintaining State Information with Hidden Fields

In our authentication example, when hidden fields are used for session management, the user authentication data are processed and the page is displayed as shown in Figure 4 upon clicking the Login button.



**Figure 4.** Using Hidden Fields

The user ID and password are stored in hidden fields as shown below:

```
<FORM NAME=myform ACTION="retrieve_hidden.jsp" method="POST">  
<input type=hidden name=userID value="peter">  
<input type=hidden name=pwd value="mypassword">  
User ID and password are stored in hidden files.<br>  
<input type=submit value="Click to retrieve hidden filed data">  
</FORM>
```

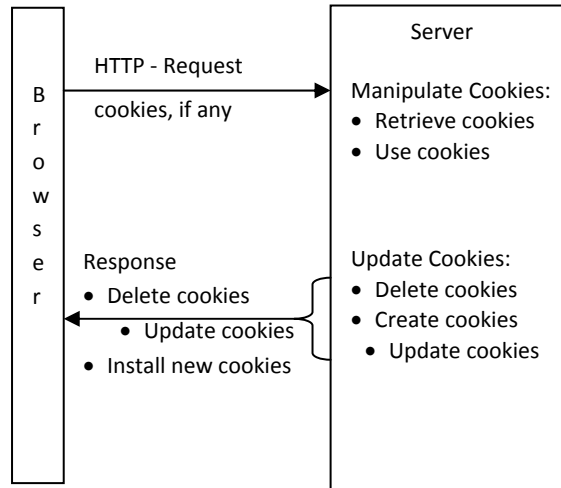
Due to simple implementation and widely supported by almost all browsers, many web applications use hidden fields to carry data among multiple requests [1]. A variety of information can be stored into hidden fields. However, if a user views source code from the browser, the user can see both those hidden fields and their values. Students can see that data stored in hidden fields are viewable, so students will understand why they should avoid using hidden fields to store sensitive data, unless they set up some way to overcome this visibility issue [2].

Furthermore, the data carried by a hidden field is very limited. It takes name/value structure and offers a single value field in which to place information. For example, `<input type=hidden name=productID value="PID20001">`. To store multiple values, students must implement delimited strings and the code to parse those strings. However, because all hidden fields are stored in the page itself, storing large values will increase the page size and slow down the response time from the server.

Hidden fields cannot carry data if a page is processed in response to a link or HTTP GET method. If a hidden field is not included in a request, the field and its value will be lost permanently and there is no way to get this value back in successive requests. If you close a browser window, there is no way to retrieve the data included in hidden fields.

## Cookies

Cookies have been widely used in websites to keep track of visitors [3]. A cookie is a small piece of textual information that a Web server sends to a browser. The browser saves the cookie to a file on your hard drive or temporary memory. The browser returns the cookie unchanged to the server when a request to access the web page is sent to the server. Thus cookies can be used to preserve knowledge of the client browser across multiple pages over periods of time. Figure 5 illustrates the use of cookies to preserve data for multiple requests.



**Figure 5.** Maintaining State Information with Cookies

The following JSP code shows how user authentication information is stored in a client machine as cookies and the cookies' life time is set to one hour.

```
<%  
String userID=request.getParameter("userID");  
String pwd=request.getParameter("pwd");  
Cookie userID_Cookie=new Cookie("userID",userID);  
Cookie pwd_Cookie=new Cookie("pwd",pwd);  
userID_Cookie.setMaxAge(60*60);  
pwd_Cookie.setMaxAge(60*60);  
response.addCookie(userID_Cookie);  
response.addCookie(pwd_Cookie);  
%>
```

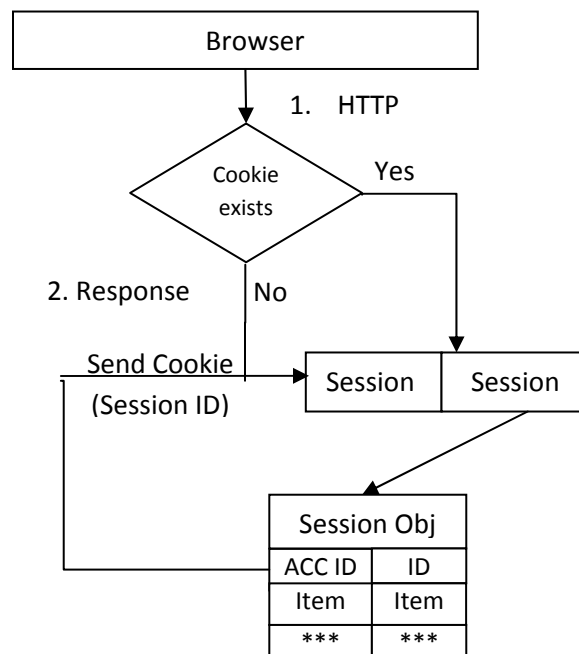
Since a cookie has a single value, the data carried by a cookie is limited. Cookies can be created to contain expiration information and may last beyond a single interactive session. Such cookies are referred to as "persistent cookies", and are stored on the client browser's hard-drive in a location defined by the particular browser or operating system. By omitting expiration information from a cookie, the client browser is expected to store the cookie only in memory. These "session cookies" will be erased when the browser is closed.

Unlike "Query String" and "Hidden Field", cookies cannot be viewed from UTL or "View Source" page. However, cookies are stored as textual format on a client machine. Therefore any user of the computer may view cookies. Revealing the information stored in cookies can be a security problem, if the cookies contain sensitive data. To

tackle this security issue, students are taught how to encrypt cookies [4]. Because cookies are stored on a client machine, no server resources are required.

### Session Object

Managing state across multiple requests through the use of a unique session ID is the most popular method for session management. In this case, a session object, which is uniquely associated with a particular client request, is created in the server when a client makes the first request to access a website. Then a unique session ID associated with this session object will be sent to and installed in the client machine as a cookie. The session ID (cookie) serves as bridge to access the session object created for this client in the successive requests. Every time the user tries to access certain pages, this cookie is sent to the server. By using the cookie information, the server identifies the user, retrieves the user's session object, and thus maintains the user's state information. This technique of using cookies to store session ID information is referred to as cookie-based session management. Figure 6 shows how data across multiple requests are shared.



**Figure 6.** Session Management With Session Object

In this process, a session object is created when a client makes the first request to access a website. A session ID used to identify the session objects is sent and installed in the browser as a cookie. During successive requests, the cookie containing the value of the stored session ID will be sent along with the requests. Upon receiving a request from the browser, the server retrieves the session ID and uses this ID to locate its corresponding session object. Therefore, the state information stored in its corresponding session object can be accessed during successive requests. The following code segment illustrates how to store user id and password in a session object:

```
<%  
String userID=request.getParameter("userID");  
String pwd=request.getParameter("pwd");  
session.setAttribute("userID",userID);
```

```
session.setAttribute("pwd",pwd);  
%>
```

Cookie-based session management provides the easiest way to manage the state information across sessions. However, students should be aware that it works only when a browser accepts cookies. If the browser is set to block cookies, this approach fails. In our class, students are further led to discuss the difference between using cookies to maintain the state information and using the cookie-based session object for the session management. In cookie-based session management, data are bound to a session object and the session object resides in the server, not the client machine while only one cookie storing the session ID is installed on the client machine. When cookies are used to share data across multiple requests, the data are stored in cookies installed in the client machine. Thus, when a browser requests a page, all those cookies are sent back to the server for processing. Compared to other cookies, the cookie used to access the session object has a shorter life and expires when the user logs off or closes the browser. In addition, the cookie storing session ID is created automatically by the application server. Like all other cookies, such as cookies to store user id and password discussed in the previous section, they all must be explicitly created and processed by using server side scripts, which is JSP in our example.

### Use URL Rewriting

Using session objects requires the cookie support function from browsers. If cookies are unsupported by the browsers, one solution is to maintain the session ID by URL rewriting. With this technique, the session ID is appended to the end of each encoded URL [5], and the session ID is sent to the server as part of the client request. On the server side, the session ID is used to access its corresponding session object, where the state information is stored. Figure 7 illustrates how URL rewriting is used to maintain the state information.

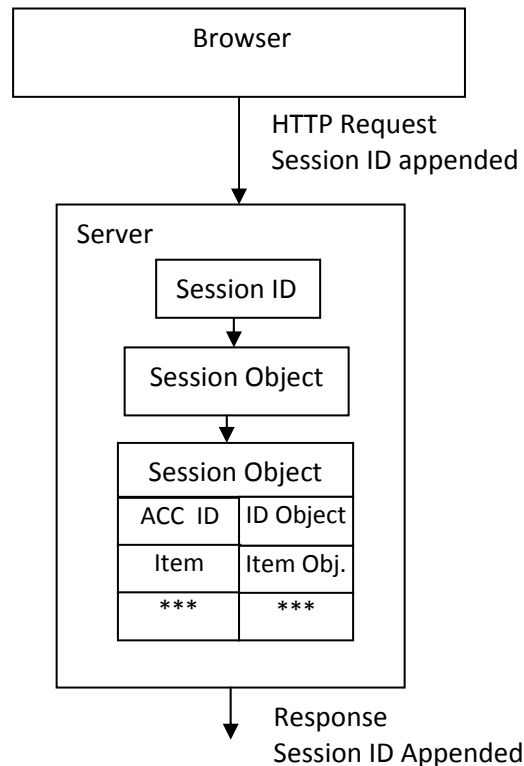
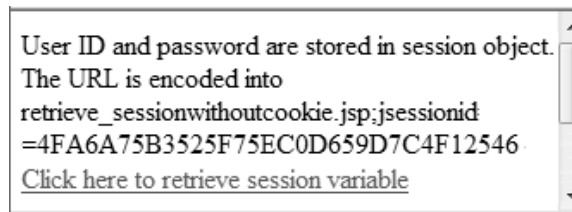


Figure 7. Session Object with URL Rewriting

Similar to the technique presented in the previous section, the session ID with URL rewriting is used to access the session object. The only difference is how the session ID is stored on the client browser. The cookie based session object management stores the session ID in a client's temporary memory as a cookie while the URL rewriting stores the session ID by appending the information to the end of each URL and passes it back and forth between the server and the client. The following code segment shows how to rewrite URL in JSP:

```
response.encodeURL("retrieve_sessionwithoutcookie.jsp");
```

If a browser blocks cookies, then the above code will generate URL rewriting as shown in Figure 8 with session ID attached to it.



**Figure 8.** An Example of URL Rewriting

Even though URL rewriting provides a solution to use a session object for session management, students should be aware that URL rewriting is not supported by all kinds of application servers. Another disadvantage is that URL rewriting requires rewriting all the URLs within a web application, a tedious task.

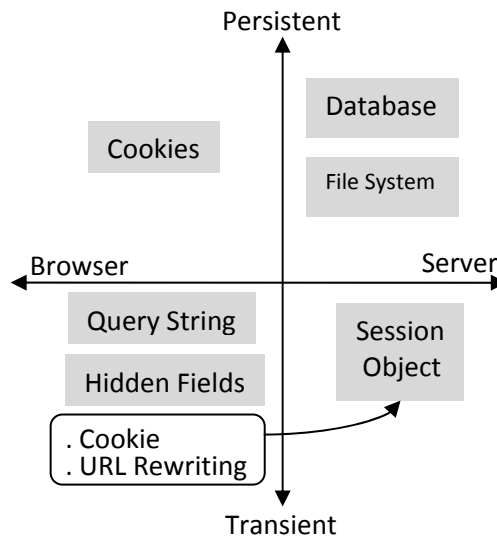
Both cookie-based session management and URL rewriting session management store the state information in a session object that resides in the server memory. A session object is constructed with expiration information. A session object will be removed from memory when it expires or the server is powered off. Therefore, data bound to a session object are accessible only when the session object is alive in the server memory. Students should be aware that storing a large amount of data in a session object could be a problem for sites with a large user base. For example, if 5 kilobytes (KB) of data are stored in each user's session and there are 1000 users with active sessions, it takes 5 megabytes (MB) of memory in a combination of physical memory and virtual memory. If there are a million active sessions, then 5 gigabytes (GB) of memory are required. Based on the available hardware, the amount of data stored in the session has a direct impact on the number of simultaneous users that can practically be supported.

A common approach to reducing memory utilization of session data is to store only critical and frequently used data in the session, for example, a username or an ID number. Actual session-specific (client related) data is stored in some other repository, such as a database or a server file system, which can be accessed as needed using the reference information residing in the session as the key for restoring it from the repository. For example, a user ID can be used to retrieve a user record from a database.

We provide a summary to students. Each session management technique provides a different mechanism to share the state information across multiple requests. Some of them store data in the server, and the others store data in client machine; some of them provide persistent data that exist for a longer period than data stored by other session techniques and are kept even after the browser is closed or the machine is powered off. Figure 9 summarizes the persistency and the location of state data of each session management technique.

Database and file system can be used in conjunction with all session management techniques we discussed here. All techniques can be used to carry authentication information to uniquely identify a client, which in turn can be used to access a file system or a database. In this way, the actual state information is stored in either file system or a database [6].





**Figure 9.** Persistency and Location of Shared Data

### IMPLEMENTATION OF SESSION MANAGEMENT TECHNIQUES IN A WEB APPLICATION

Session management techniques provide mechanisms to hold some information that is particular to a client. Most Web applications could not be developed without session management. A Web application that facilitates access to confidential information through login would allow its users to access this data through a single login process. This feature is made possible through implementing session management techniques in the application. Otherwise, a user who accesses a page containing user-specific data would be required to login every time when this user makes a request. There are many Web applications having a “remember me” functionality, where return visitors are automatically logged in or presented with custom content. For instance, Yahoo email allows you to auto log into your email account for 2 weeks if you check “Keep me signed in” checkbox when you sign in to your yahoo email account. Given the nature of stateless HTTP protocol, web application developers have to use one of these session management techniques to uniquely track users and share certain information throughout a web application. Choosing among the options for the available state management techniques depends heavily upon the nature of the application, and it should be made by students based on the following criteria:

- How much information do you need to store during successive requests?
- When does the stored information need to be accessed?
- How long do you need the information to be available for retrieval?
- Does the client accept persistent or in-memory cookies?
- Where do you want to store the information: on a client machine or on the server?
- Is the information sensitive?
- What sorts of performance criteria do you have for your application?

If a Web application needs a small amount of data to be carried over through the application, the state information can be managed with any of these techniques. If a large amount of data is needed to maintain the state information, only authentication information may be managed by the session management techniques, and the real state data should be stored in either a database or a file system. The following table shows students the factors that affect the implementation of session management.

**Table 1.** Comparison of Session Management Techniques

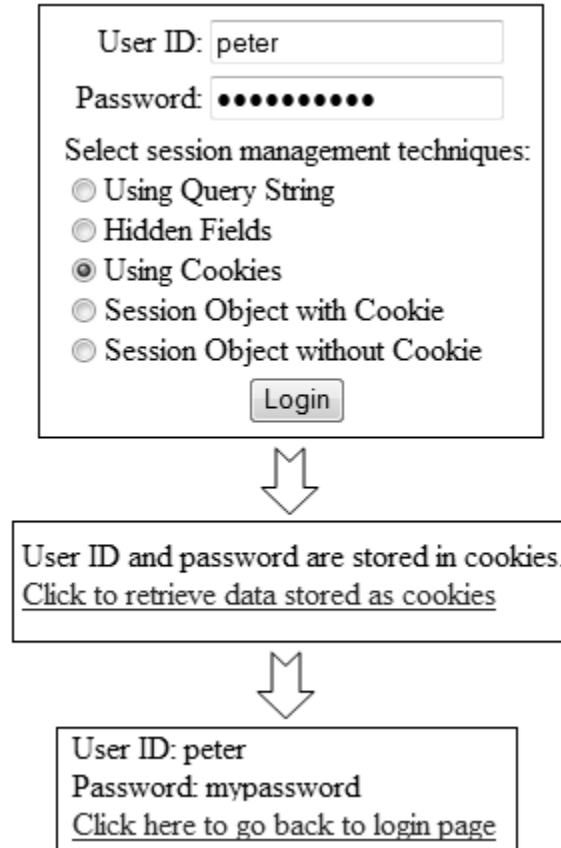
Features	Session Management Techniques				
	Query String	Hidden Fields	Cookies	Session Objects	URL Rewriting
Life Time	Book mark History	Browser	Fixed upon creation	Fixed at server configuration	Fixed at server configuration
Where Data Stored	Browser	Browser	Client Hard Drive	Server Memory	Server Memory
Security	Least Secure	Moderate	Moderate	High	High
Accessibility	Client/Server	Client/Server	Client/Server	Server	Server
Data Types	Textual	Textural	Textural	Objects	Objects
Need Cookie Enabled	No	No	Yes	Yes	No
Amount of Data to Share	Small	Small	Moderate	Moderate	Moderate

Many web applications use hidden fields to carry information for successive requests, especially if only a small amount of data is involved. A typical use of hidden fields is to carry information from page to page when you sign up a new account for a website. In this process, user's information to be collected is split into several pages. And the data collected from each page may be carried over by using hidden fields until the client submits the last page to processing. However, storing the state information with hidden fields may be not appropriate if a large amount of data is involved. First, every page must be dynamically generated to include new hidden fields. Second, the page request carrying hidden fields cannot be interrupted by closing the browser or visiting other pages that do not carry hidden fields. Otherwise, all hidden fields may be lost. Cookies can be used to store and retrieve the state information in successive requests. However, using cookies requires the cookie support function of the browser. The session object has been widely used for the session management. However, since the session object uses server resources, server's performance may be jeopardized if too many session objects are kept in the server memory.

Even though all these techniques can be used for session management to store state information, some of them are not suitable for certain circumstances. If an implementation requires persistent data to be stored in a client machine, then a cookie is the only way to store the state information to meet the requirement. For example, "remember me" functionality, auto login, customized web page, and searching preference all require the state information to be stored in the client machine as persistent cookies. Persistent cookies cannot be implemented by any of these session management techniques other than cookies.

#### **EXERCISE AS PEDAGOGICAL TOOL**

To stimulate a deep understanding, we designed a simple authentication web application as shown in Figure 1 and required students to experiment with each of the session management techniques discussed in this paper. The login page contains two input fields, one for user ID and the other one for password. The five techniques are listed as radio buttons so that students can choose only one method each time. There are five separate web pages; each of them is implemented with one of the session management techniques. The data collected on the login page is processed and stored by using the method selected on the login page. Then there is a link on the page to retrieve the shared information. The following figure shows the flow chart of the process:



**Figure 10.** Flow Chart for the Experiment of Session Management Techniques

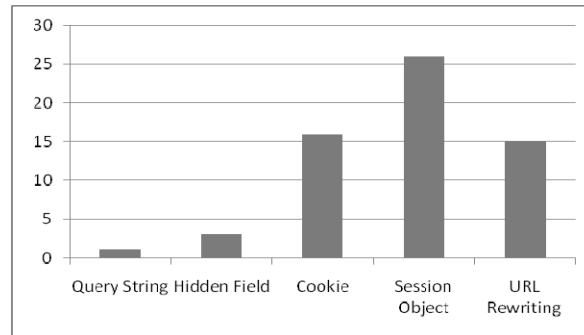
To investigate how the security setting in browser will affect the implementation of session management, students were directed to check the browser security setting to make sure the browser to accept cookies. Server side scripts were written in JSP and source codes were distributed to students as hardcopies. All students were required to experiment with each of the methods and then interact with students on the following learning experience:

- (1) Discuss where the user id and password information are stored in each of the session management techniques and how the location of the information affects the use of the client machine and the server.
- (2) Discuss whether the user id and password are viewable on browser window, either from URL or by viewing the “View Source” option of the browser. And then discuss how this issue is related to your security concern.
- (3) Discuss advantages and disadvantage of each technique in terms of easiness of implementation and security.
- (4) Discuss how each technique affects the Internet traffic, the usage of client computer memory, and the usage of server machine’s memory.
- (5) Discuss if there is any difference between “Session Object” and “URL Rewriting” techniques if a browser accepts cookies.

After experiment with cookie enabled browser setting, students are instructed to change browser security settings to block all cookies and repeat the experiment. And then:

- (6) Discuss how the security settings affect the implementation of those session management techniques.

At the end of the experiment, students were asked to vote for their favorite session management techniques. The results are summarized in Figure 11.



**Figure 11.** Favorable Session Management Techniques

## STUDENT FEEDBACK

The class resulted in the following learning outcomes. Students said that the exercise and discussion:

- (1) Helped them understand the concept of stateless of the HTTP protocol and how to overcome the limitation of the protocol with session management techniques.
- (2) Helped them understand the session management techniques and the differences among those techniques.
- (3) Helped them understand how a browser's security setting will affect the implementation of session management.
- (4) Helped them understand how a session management technique could expose security problem if not implemented properly.

## CONCLUSION

The concept of state, or the ability to remember information as a user travels from page to page within a website, is important for designing Web applications. The HTTP protocol is stateless in the sense that it does not remember any information being processed in the previous request. On the other hand, many Web applications, like shopping carts, accessing email, online banking, accessing Blackboard course management systems, and all other applications requiring identity verification, require some state information to be stored and available to the Web applications. Thus, developers of Web applications must take it upon themselves to code the state information by applying one of the session management techniques.

This paper examined and compared five commonly used session management techniques: query string, hidden field, cookies, session object, and URL rewriting. By providing a detailed discussion on each of these session management information plus a hands-on exercise, students gained considerable insight into the differences among these session management techniques, and they are able to choose an appropriate session management technique for their course projects in the Web application development course.

## REFERENCES

1. Andrews, M., & Whittaker, J. (2006). How to break web software: functional and security testing of web applications and web services. *Boston, Ma: Addison-Wesley.*
2. Cao, Q., Davis, J. S., Bai, X., & Katter, O. E. (2002). Using ASP-based message encryption project to teach information security concepts. *Journal of Information Systems Education, 13*(3), 183-187.
3. Doran, A. (2007, September 10). Web services now in session. *InformationWeek.*

4. Gutzmann, K. (2001). Access control and session management in the HTTP environment. *IEEE Internet Computing*, 5(1), 26-35.
5. Bangalore, K. (2009). HTTP session management in weblogic. Retrieved from <http://cafe-obabe.blogspot.com/2009/03/http-session-management-in-weblogic.html>
6. Tels, B. (2008, October 29). Rethinking session management. Gridshore. Retrieved from <http://www.gridshore.nl/2008/10/29/rethinking-session-management/>