ASP.NET - State Management Techniques

- Background : Assume, someone is trying to access a banking website and he has to fill in a form.
- So the person fills in the form and submits it.
- After submission of the form, the person realizes he has made a mistake.
- So he goes back to the form page and he sees that the information he submitted is lost.
- So he again fills in the entire form and submits it again.
- This is quite frustrating for any user.
- So to avoid such problems "STATE MANAGEMENT" acts as a savior for developers like us.
- If you do not understand what the word "STATE" means, then think about it, as some interaction between the User and the Server.

ASP.NET - State Management Techniques

State management is the technique or the way by which we can maintain/store the state of the page or application until the User's Session ends.

State management is the process by which ASP.NET let the developers maintain state and page information over multiple request for the same or different pages.

ASP.NET - State Management Techniques

- Generally, web applications are based on stateless HTTP protocol which does not retain any information about user requests.
- A new instance of the Web page class is created each time the page is posted to the server.
- In traditional Web programming, this would typically mean that all information associated with the page and the controls on the page would be lost with each round trip.
- For example, if a user enters information into a text box, that information would be lost in the round trip from the browser or client device to the server.
- To overcome this inherent limitation of traditional Web programming, ASP.NET includes several options that help you preserve data on both a per-page basis and an application-wide basis.

Types of State Management

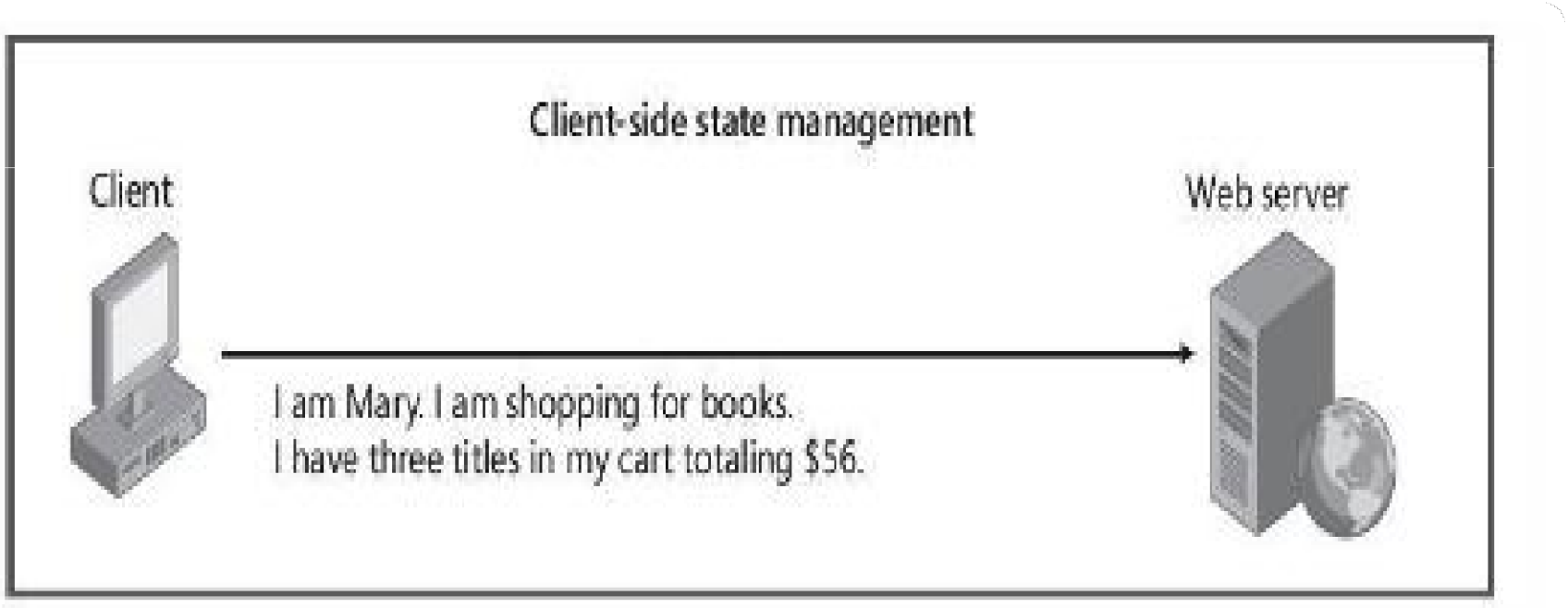
- There are mainly two types of state management that ASP.NET provides:

  Client side state management

  Server side state management

Client-side State Management

- When we use client side state management, the state related information will be stored on client side.
- This information will travel back and forth with every request and response.

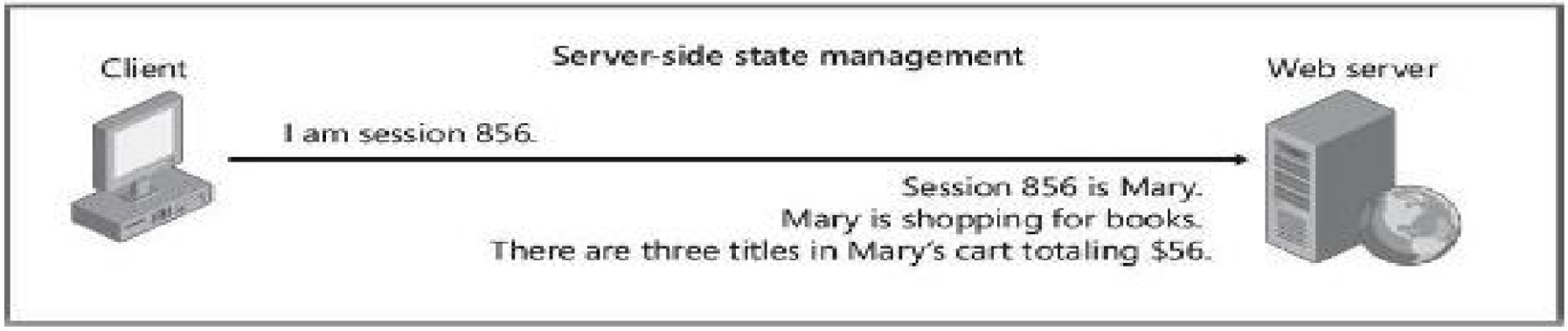


Client-side State Management

- The major benefit of having this kind of state management is that we relieve the server from the burden of keeping the state related information, it saves a lot of server memory.
- The downside of client side state management is that it takes more bandwidth as considerable amount of data is traveling back and forth.
- But there is one more problem which is bigger than the bandwidth usage problem.
- The client side state management makes the information travel back and forth and hence this information can be intercepted by anyone in between.
- So there is no way we can store the sensitive information like passwords, creditcard number and payable amount on client side, we need server side state management for such things.

Server-side State Management

- Server side state management, in contrast to client side, keeps all the information in user memory.
- The downside of this is more memory usage on server and the benefit is that users' confidential and sensitive information is secure.
- Server-side options for storing page information typically have higher security than client-side options, but they can use more Web server resources, which can lead to scalability issues when the size of the information store is large.



Various State Management Techniques

Client-Side

Stores data on the client in various ways.
Bandwidth should be considered while implementing client side state management options because they involve in each roundtrip to server.
Example: **Cookies are exchanged between client and server for each page request.**

Server-Side

Stores data in memory on the server.
you can decrease the amount of information sent to the client in order to preserve state, however it can use costly resources on the server.

Various State Management Techniques

Client-Side

- View State
- Control State
- Hidden Fields
- Cookies
- Query String

Server-Side

- Application State
- Session State
- Profile Properties

ViewState

- ViewState is the mechanism that allows state values to be preserved across page postbacks.
- Because of the stateless nature of web pages, regular page member variables will not maintain their values across postbacks.
- When we need a page variable to maintain its value across page post backs, we can use ViewState to store that value.
- Values stored in ViewState will be serialized and sent to the client browser as the value of a hidden form input.
- When you view the page source (in your browser) of a page the uses ViewState, you may see this hidden viewstate input which will look something like this:

```
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="/wEPDwUKMTM1ODM3Nj....." />
```

ViewState

- When the page is processed, the current state of the page and controls is hashed into a string and saved in the page as a hidden field, or multiple hidden fields if the amount of data stored in the ViewState property exceeds the specified value in the MaxPageStateFieldLength property.
- When the page is posted back to the server, the page parses the view-state string at page initialization and restores property information in the page.
- This single hidden field contains all the viewstate values for all the page controls.
- Because viewstate is (by default) sent to the client browser and then returned to the server in the form of a hidden input control on your page, storing a significant amount of data in viewstate can increase your page size and can affect your page performance.

ViewState

- The ViewState property provides a dictionary object for retaining values between multiple requests for the same page.
- This is the default method that the page uses to preserve page and control property values between round trips.
- You can store values in view state as well.
- View State can be used to store state information for a single user.
- To disable ViewState for a control, you can set the EnableViewState property to false.
- When ViewState is disabled for any control, it will also automatically be disabled for all child controls of that control.

ViewState

- Example of ViewState:

```
// Add item to ViewState
    ViewState["myviewstate"]  = myValue;
//Reading items from ViewState
    Response.Write(ViewState["myviewstate"]);
```

- **Advantages of ViewState:**
  - No server resources are required
  - Simple implementation
  - Enhanced security features
- **Disadvantages of ViewState:**
  - Performance considerations
  - Device limitations
  - Potential security risks

Summary of client-side SMTs

State Management option	Recommended usage
View state	Use when you need to store small amounts of information for a page that will post back to itself. Using the ViewState property provides functionality with basic security.
Control state	Use when you need to store small amounts of state information for a control between round trips to the server.
Hidden fields	Use when you need to store small amounts of information for a page that will post back to itself or to another page, and when security is not an issue. You can use a hidden field only on pages that are submitted to the server.
Cookies	Use when you need to store small amounts of information on the client and security is not an issue.
Query string	Use when you are transferring small amounts of information from one page to another and security is not an issue. You can use query strings only if you are requesting the same page, or another page via a link.

Summary of server-side SMTs

State Management option	Recommended usage
Application state	Use when you are storing infrequently changed, global information that is used by many users, and security is not an issue. Do not store large quantities of information in application state.
Session state	Use when you are storing short-lived information that is specific to an individual session and security is an issue. Do not store large quantities of information in session state. Be aware that a session-state object will be created and maintained for the lifetime of every session in your application. In applications hosting many users, this can occupy significant server resources and affect scalability.
Profile properties	Use when you are storing user-specific information that needs to be persisted after the user session is expired and needs to be retrieved again on subsequent visits to your application.