Web Technologies

Lecture 6

State preservation

Motivation

- How to keep user data while navigating on a website?
  - Authenticate only once
  - Store wish list or shopping cart items while browsing an online shop
  - Remember user preferences when displaying a page

Stateless vs. stateful

- **State** – a set of conditions at a moment of time
  - Computers are inherently stateful in operation
- Describe whether or not a computer is designed to note and remember one or more preceding events in a sequence of interactions
- **Stateful** means that a computer keeps track of the state of interaction
- **Stateless** means that no record of previous interactions are kept and that each interaction request is handled solely based on information that comes with it

Sessions

- A **semi-permanent** interactive **information interchange**
- Set up or established at a **certain point in time**
- Basic requirement to perform **connection-oriented** communication
- **Enables** stateful communication

Stateless protocol

- Protocol that treats each request as an **independent** transaction
- Communication consists of a paired request-responses
- It does not require the server to retain session information
- Examples
  - IP
  - HTTP

Stateful protocol

- Requires **keeping** the internal state on the server
- Examples
  - FTP
    - During a session the user provides authentication details and sets various variables
    - All details are stored on the server as part of the user state

Pros and cons

Advantages of **stateless** communication

- Simplifies the server design
- No need to dynamically allocate storage
- If client dies in mid-connection no need to clean up the state

However

- Requires additional information in every request
- The information needs to be processed on the server

Stateful HTTP

- Keep information between different requests
- Useful in many cases
  - Stores user information when navigating a website
    - Authentication credentials
    - Shopping cart items
    - Search preferences
- HTTP is stateless → need artificial constructs
  - Hidden form variables
  - HTTP Cookies
  - Web Storage (HTML 5)
  - Server side session variables
  - URL rewriting using URI-encoded parameters

Client side web sessions

- State information is kept on the client
- Approaches
  - **Hidden variables**

    <input type="hidden" name="userName" value="John Doe">
  - **Cookies**
    - Format: *cookieName=cookieValue*
    - Handled using Javascript

      document.cookie="username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC";

Data flow

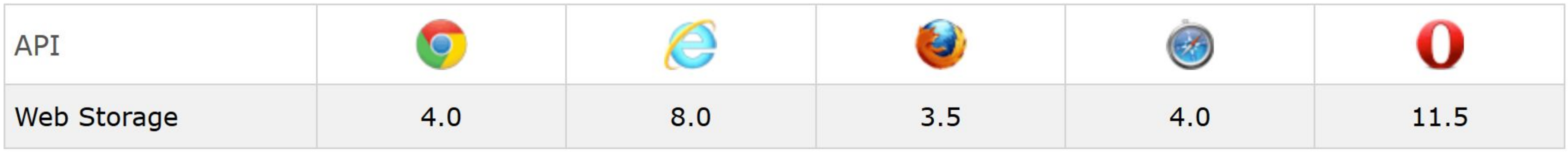
1. Server sends current state to client
2. Client stores state in a cookie
   - In memory
   - On disk
3. For each successive request client sends cookie information to server
4. Server uses cookie data to remember the state of the web application

Client side issues

- Prone to tempering from user or locally installed software
- When confidentiality and integrity is required
  - Only the server must be able to interpret the data
  - Only the server should manipulate data
  - Only the server should initiate valid sessions
  - Encryption is required
- Cookies should be small to avoid communication overhead
  - Data compression may be needed for large session data
- Logout not fully implemented
  - Clients can drop cookies but data can be resent by the server

Web storage

- **Alternative** to cookies
- Implemented in HTML 5
- **Advantages**
  - Security
  - Can store more data than a cookie (>5Mb)
  - Information is never transferred to the server
  - Local storage is per origin
    - All pages from one origin can store and access the same data



API					
Web Storage	4.0	8.0	3.5	4.0	11.5

Using web storage

- **localStorage** object

```
localStorage.setItem("lastname", "Smith");
var name = localStorage.getItem(“lastname”);
localStorage.removeItem("lastname");
```

- **sessionStorage** object
  - Similar methods to localStorage
  - It keeps data only for the current session
    - If the tab is closed data is lost

Server side web sessions

- Full control of the session
  - Can terminate a session on demand
- Existing frameworks can reduce the amount of code to handle sessions
  - Apache Shiro
- Can handle larger data than a cookie
- Only reference to session ID is sent over HTTP as a cookie
- Implementation can change independent on client

Server side issues

- More points of failures
  - If DB is down sessions cannot by created, updated, or validated
- More overhead in handling sessions
  - Requires asynchronous DB write
- Web applications can only verify a session by communicating with the server

What’s next?

- AJAX
  - Synchronous vs. asynchronous
- JQUERY
- Server side programming
- Web services
- Cloud computing