Basics of Web Applications

Conventional (Old-time) Communication

1. Client SW (C) connects to server SW (S) to receive some info.

2. S sends requested info to C.

3. S establishes a *state* of C (i.e., everything C has said) in its memory.

4. Connection is closed.

Definition of *State*

Definition: The set of the current values of all variables (including the local, dummy variables) of a SW component (e.g., a function (FOP) or an object (OOD)) at a certain point of time *t* is called the *state* of the SW component.

HTTP (HyperText Transfer Protocol)

- The basic tool for communication on the web
- The most important thing to know about HTTP is that it is stateless
- *stateless* (why?) (what does it mean?)
  - Millions of clients seeking access to some servers
  - No information about client stored.
- Every time you link to a web site, you make an independent HTTP request.

A typical Flow of an HTTP Session

- Type www.yahoo.com into browser.
- Browser translates web page address into an IP address and establishes a connection with the address (thru port 80). The bidirectional data transfer protocol is "***T***ransmission ***C***ontrol ***P***rotocol" or TCP.
- Once connection is established, browser specifies its request (GET, POST) using a specific format such as a file name, the preferred protocol etc.
  - Ex: browser transmits "GET / HTTP/1.0"
    - GET means a file is requested.
    - The name of file follows (in our case it is "/", the root index page)
    - "HTTP/1.0" specifies that browser would like a response in HTTP 1.0 protocol.

HTTP Session cont'd...

- *yahoo returns* a set of *headers* stating the *protocol* used, *whether* or not the requested *file* is *found*, *size* of file and *type* of information in the file (the MIME type).
- A *blank line* follows to indicate end of headers
- *Contents of file* are transmit following the blank line.
- TCP *connection is closed* when browser entirely receives the file.

Discussion

- Scenario: You are filling out an application form to be a member in www.flickr.com. Right before filling the last field after a five-minute work, your friend drops by for a conversation. Half an hour later, when you are back in front of your monitor to fill in the last field and complete your application you see that the connection is lost.
- What would you expect?
- How would a *stateless* server behave?
- Problem(s)?
- Suggestions...

Challenge

To create a *stateful behavior* of a *stateless SW system!*

Stateful Behavior

- For a stateless SW system to display stateful behavior,
  - it has to identify the user,
  - save contents of user's session, and
  - recall them if they are requested within a reasonable span of time.

Solution Alternatives?

- Keep information in a log file indexed by name on web server
  - Problem: web servers do not know the user but know the IP address of user's computer.
  - indexing log file by IP address will not make the info. user-specific.
- How about attach some ID info. (*session ID*) to a user that will be returned on that user's next request?

Cookies

- The first paragraph of Netscape's “Persistent Client State HTTP Cookies – Preliminary Spec.”:
- *Cookies are a general mechanism which server side connections can use to both store and retrieve information on the client side of the connection. The addition of a simple, persistent, client-side state significantly extends the capabilities of Web-based client/server applications.*

Cookies

- An extension to HTTP.
- A tool on the client used by the server to store and retrieve information (i.e., client side *state*)
- Distributed DBMS?...

Problems with Cookies

- Limited space for cookies at client
  - e.g., store at most 20 cookies 4 kbytes each!
  - for an extensive web search not enough?...
  - even these limits may have the server slow down web interaction, especially on-modem users!
- Not portable for users!
  - A user intending to start with a web session at one computer and continue at another cannot.
- Cookies may be rejected due to privacy issues.
  - Check the figure 2.2 from the text book for an example of privacy issues

Solution Alternatives

- Two types of cookies:
- First cookie:
  - Adds a browser id
  - Keeps browser-specific information such as
    - quality of communication (e.g., text-only)
    - language preferred
  - Lasts longer (e.g., years)
- Second cookie:
  - Adds a session id
  - Keeps session-specific information
  - Lasts much shorter (e.g., hours or several days at most)

Server Side Storage

- A storage for keeping diverse contents of user sessions within the Internet environment needs to handle concurrent read and write accesses of multiplicity of users.
- A good solution handling the concurrent access problem is the DBMSs.

Properties of a Sufficient DBMS

How do we evaluate whether or not a DBMS is powerful enough for our application?

A good DBMS should pass the so called ACID test

- Atomicity
- Consistency
- Isolation
- Durability

Atomicity

Results of a transaction's execution are either all committed or all rolled back. All changes take effect, or none do.

Suppose that a user is registering by uploading name, address and JPEG portrait into three separate tables. A Web script tells the database to perform three inserts as part of a transaction. If the hard drive fills up after the name and address have been inserted but before the portrait can be stored, the changes to the name and address tables will be rolled back.

Consistency

A transaction is legal only if it obeys user-defined integrity constraints. Illegal transactions are not allowed and, if an integrity constraint cannot be satisfied, the transaction is rolled back.

For example, suppose that you define a rule that postings in a discussion forum table must be attributed to a valid user ID. Then you hire someone to write some admin pages. He writes a delete-user page that does not bother to check whether or not the deletion will result in an orphaned discussion forum posting. An ACID-compliant DBMS will check, though, and abort any transaction that would result in you having a discussion forum posting by a deleted user.

Isolation

The results of a transaction are invisible to other transactions until the transaction is complete.

For example, suppose you have a page to show new users and their photographs. This page is coded in reliance on the publisher's directive that there will be a portrait for every user and will present a broken image if there is not. Jane Newuser is registering at your site at the same time that Bill Olduser is viewing the new user page. The script processing Jane's registration has completed inserting her name and address into their respective tables. But it is not done storing her JPEG protrait. If Bill's query starts before Jane's transaction commits, Bill won't see Jane at all on his new users page, even though jane's insertion into some of the tables is complete.

Durability

Once committed (completed), the results of a transaction are permanent and survive future system and media failures.

Suppose your e-commerce system inserts an order from a customer into a database table and then instructs CyberSource to bill the customer $500. A millisecond later, before your server has heard back from CyberSource, someone trips over the machine's power cord. An ACID-compliant DBMS will not have forgotten about the new order.

Furthermore, if a programmer spills coffee into a disk drive, it will be possible to install a new disk and recover the transactions up to the coffee spill, showing that you tried to bill someone for $500 and still are not sure what happened over at CyberSource. Notice that to achieve the D part of ACID requires that your computer have more than one hard disk.

Why RDBMS?

- Why is the RDBMS the dominant technology for persistence behind a Web server?
- There are three main factors:
  - The first pillar of RDBMS popularity is a **declarative** query language called “SQL”.
    The most common style of programming is not declarative; it is called “**imperative**” or “**procedural**”. This style of programs have two drawbacks:
    - Becomes quickly complex and then can be developed and maintained only by professional programmers.
    - Contain a lot of errors. Detection of these errors may be very hard.

Why RDBMS?

- The second pillar of RDBMS popularity is **isolation of important data** from programmers' mistakes. With other kinds of database management systems, it is possible for a computer program to make arbitrary changes to the data set. The RDBMS limits programmers to uttering very simple statements of the form INSERT, DELETE, and UPDATE.

- The third pillar of RDBMS popularity is **good performance** with many thousands of simultaneous users.

Internet Application Development Steps

1)Develop a Data Model.

- What information are you going to store and how will you represent it?

2)Develop a collection of legal transactions on that model,

- e.g., inserts and updates.

3)Design the page flow.

- How will the user interact with the system?
- What steps will lead up to one of those legal transactions?

4)Implement the individual pages.

- You will be writing scripts that query information from the data model, wrap that information in a template (in HTML for a Web application), and return the combined result to the user.

Referance

Greenspun, P., Andersson, E., & Grumet, A. (2006). Software engineering for Internet applications.