Session State Pattern

SWE 4601

Patterns of Enterprise Application Architecture - Martin Fowler, Chapter 3, 17

What is it?

- Stateful
  - maintains information or state about a client or user across multiple interactions.
  - online shopping sites, banking applications, and social media platforms
- Stateless
  - A stateless application is one that does not maintain any information or state about a client or user between interactions
  - simple websites, and static content delivery networks

Client Session State

- Stores session state on the client.
- How It Works?
  - Three common ways to do client session state: URL parameters, hidden fields, and cookies.

- **URL parameters** are the easiest to work with for a **small** amount of data.
- the size of an URL is limited,
- it’s a popular choice for something like a session ID

- **Hidden field**: A hidden field is a field sent to the browser that isn’t displayed on the Webpage. You get it with a tag of the form <INPUT type = “hidden”>.
- Serialization and deserialization is needed i.e., XML to object and object to XML

Client Session State

- **Cookies**, which are sent back and forth automatically. Just like a hidden field you can use a cookie by serializing the session state into it.
- Cookie handles session state as key value pair.
- Problem: many people don't like cookies and turn them off. If they do that, your site will stop working.
- Cookies also work only within a single domain name, so if your site is separated into different domain names the cookies won't travel between them.
  - Response.Cookies("UID")=1
  - Response.Cookies("UID").Domain = ".myserver.com"

Client Session State

- When to use it?
  - It reacts well in supporting stateless server objects with maximal clustering and failover resiliency. if the client fails all is lost, but often the user expects that anyway.
  - the amount of data involved. With just a few fields everything works nicely.
  - For large where to store the data and the time cost of transferring everything with every request become prohibitive
  - security issue. Any data sent to the client is vulnerable to being looked at and altered. Encryption is the only way to stop this, but encrypting and decrypting with each request are a performance burden.
  - Data alteration needs to revalidate sent back data

Server Session State

- Keeps the session state on a server system in a serialized form.
- How It Works?
  - a session object is held in memory on an application server.
  - You can have some kind of map in memory that holds these session objects keyed by a session ID; all the client needs to do is to give the session ID and the session object can be retrieved from the map to process the request.
  - Assumption:
    - application server carries enough memory to perform this task.
    - It also assumes that there’s only one application server. if the application server fails, all data will be lost.

Server Session State

- Solve memory issue: not hold in memory but instead serialize all the session state to persistent storage such as database, file.
  - Binary Serialization
  - Textual Serialization, JSON, XML
- If you store it in an application server, failover resilience will be lost and clustering is not possible
- BLOB and LOB need to be stored in database, performance will vary based on database.
- Again, If you're storing Server Session State in a database, you'll have to worry about handling sessions going away. One route is to have a daemon that looks for aged sessions and deletes them.

Server Session State

- When to use it?
  - The great appeal of Server Session State is its simplicity. In a number of cases you don’t have to do any programming at all to make this work.
  - Application server platform like Java EE, Microsoft.NET Node.js etc. gives supports.

Database Session State

- Stores session data as committed data in the database.
- How It Works?
  - When a call goes out from the client to the server, the server object first pulls the data required for the request from the database. Then it does the work it needs to do and saves back to the database all the data required.
  - at least a session ID number needs to be stored on the client and send with every request.
  - To store session state data in a database, you can create a separate table or collection to store the session data for each user. The table or collection can have a unique identifier for each user session, and columns for each piece of data that needs to be stored. When a user logs in or starts a session, the system can create a new row in the table or collection with the user's session ID and default values for each column. As the user interacts with the application and updates their session data, the system can update the corresponding row in the table or collection.

Database Session State

- When to use it?
  - The first aspect to consider with this pattern is performance. You'll gain by using stateless objects on the server, thus enabling pooling and easy clustering.
  - Time to pull data from database. You can reduce this cost by caching the server object so you won't have to read the data out of the database whenever the cache is hit.
  - The second main issue is the programming effort, most of which centers around handling session state. If you have no session state and are able to save all your data as record data in each request, this pattern is an obvious choice because you lose nothing in either effort or performance