MICROSOFT ARCHITECT
By Joydip Kanjilal, Columnist, InfoWorld
JAN 28, 2016 9:50 AM PST

How to work with Sessions in ASP.Net

Take advantage of sessions to store and manage information pertaining to a logged user's session

HTTP is a stateless protocol. This implies that every time a new request is sent from the client to the server the state information of the previous request is lost. There are several ways to store and manage state in ASP.Net. Session object is one of them, the others being Caching and Application objects.

Caching improves the application's performance by minimizing the consumption of resources in your system. You can store frequently used data or Web pages to improve the application's performance and throughput and scalability by reducing the consumption of the server's resources.

You can define session as a session of connectivity between the server and the client -- the session object holds data that correspond to a user's session. Session is a server side state management technique that is used to store user specific information in the memory for later retrieval.

Modes of storing session data

The session object is created and managed at the server side. Session storage mode determines where your session data should be stored. Session state can be stored in one of the following modes:

1. In - Process: Stored in the same ASP.Net Process
2. State Server: Stored in the some other system
3. SQL Server: Stored in the SQLServer database
4. Custom: this enables you to store session data using a custom storage provider

The In-Proc mode of storage of session data is the default mode and it is also the fastest of all the available storage modes. In this mode, the session data is stored in the server's memory -- inside the ASP.Net worker process. You should use this mode if the amount of data that needs to be stored in the session is less and if you wouldn't need the data to be persisted. It should be noted that session data stored in this mode is volatile, i.e., as soon as the session is terminated the session data is lost. So, data in the session is available as long as the session is alive.

In the State Server mode, the session data is stored in a separate process - this is called the ASP.Net State Service. In other words, session data in this mode is stored outside of the ASP.Net worker process or the application pool in IIS. Unlike the In-Proc mode, session data in the State Server mode is preserved, i.e., it's not lost after your web application is restarted.

The following code snippet illustrates how you can configure session state in your application to be stored in this mode.

```
<configuration>

  <system.web>

    <sessionState mode="StateServer"

      stateConnectionString="tcpip=IDGServer:1234"

      cookieless="false"


      timeout="20"/>

  </system.web>

</configuration>
```

The SQLServer mode of session data storage is used to persist session data of your application in the SQLServer database. Similar to the State Server mode of session data storage, the SQLServer mode also enables you to persist session data of your application across application restarts. Note that you should ensure that the ASP.Net session state database is created before using this mode. You can create this database using the Aspnet_regsql.exe command line utility.

The following code snippet shows how you can configure your application to store session data in a SQLServer database.

```
<configuration>
  <system.web>
    <sessionState mode="SQLServer"
       sqlConnectionString="data source=server;user id=joydip;password=sa1@3"
       cookieless="false" timeout="20" />
  </system.web>
</configuration>
```

Support for Session data compression

Microsoft's ASP.Net 4 introduced a new feature: session state compression. With ASP.Net 4 and onward, you can leverage this built-in feature to compress session data for storing out-of-process sessions. To take advantage of this feature, all you need to do is set the compressionEnabled attribute to "true" in the configuration file of your application. The following code snippet illustrates how this can be configured.

```
<sessionState
  mode="SQLServer"
  stateConnectionString="some connection string..."
  compressionEnabled="true"/>
```

Session state enables you to store user specific data in the memory and identify a particular request uniquely. Session data is stored as key/value pairs in the SessionStateItemCollection and can be accessed using the HttpContext.Session property.

The following code examples show how you can store and retrieve session data.

```
HttpSessionState.Session["UserName"] = "John"; //stores session data

string str = HttpSessionState.Session["UserName"].ToString();

// Retrieves session data

HttpSessionState.Remove("Key to remove");

//Removes an object from the session state
```

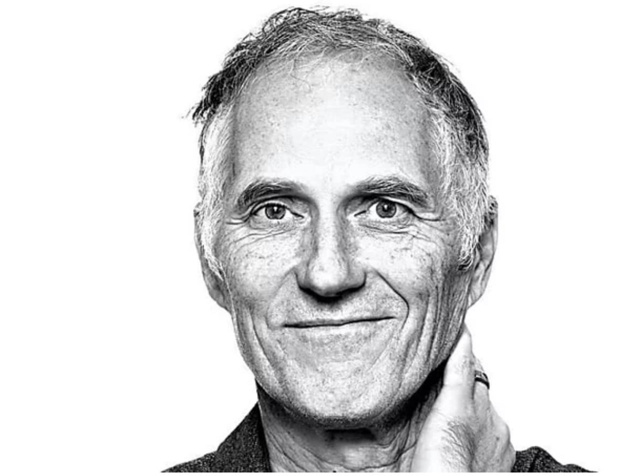
Joydip Kanjilal is a Microsoft MVP in ASP.Net, as well as a speaker and author of several books and articles. He has more than 20 years of experience in IT including more than 16 years in Microsoft .Net and related technologies.

Follow

YOU MAY ALSO LIKE

UNITED STATES ▾

Recommended by



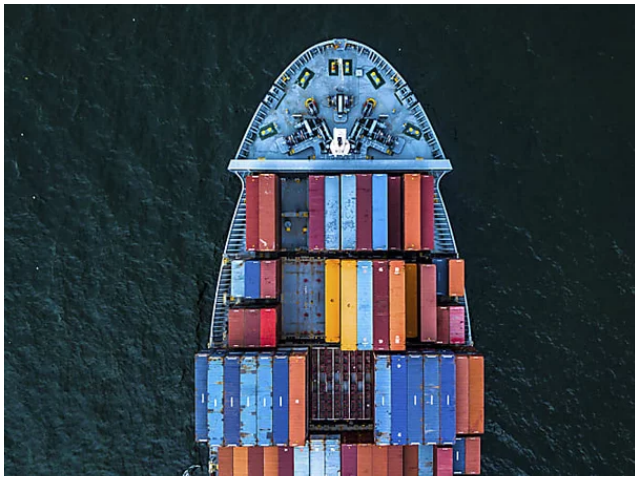
Tim O'Reilly: the golden age of the programmer is over

2 cloud architecture problems are still unsolved

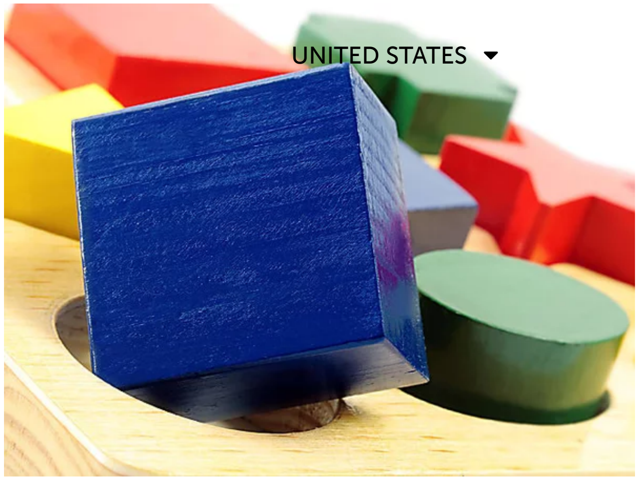
Visual Studio Code 1.54 runs on Apple Silicon

When to use Task.WaitAll vs. Task.WhenAll in .NET

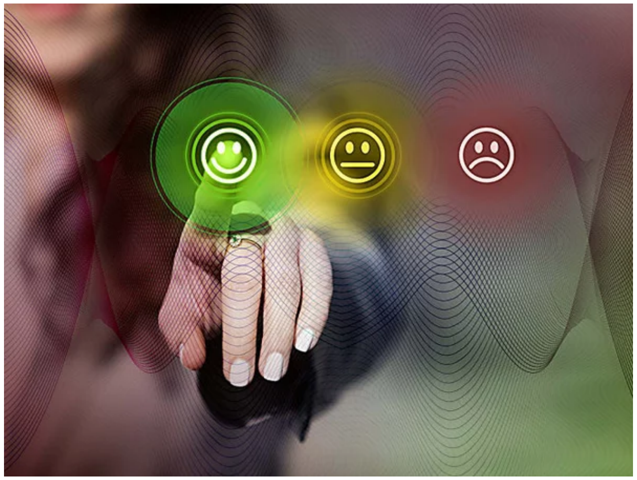
Why developers should use graph databases



What’s new in Kubernetes 1.20



What Python’s new pattern matching can do for you



Devops salaries continued to rise during the pandemic

Nominate yourself for the 2021 Enterprise Architecture Awards

6 ways Alibaba Cloud challenges AWS, Azure, and GCP

2 common cloud computing predictions for 2021 are wrong