Perry Skountrianos Microsoft

Mar 23 2019 05:1

ASP.NET Session State with SQL Server In-Memory OLTP

First published on MSDN on Nov 28, 2017

ASP.NET session state enables you to store and retrieve values for a user as the user navigates the different ASP.NET pages that make up a Web application. Currently, ASP.NET ships with three session state providers that provide the interface between Microsoft ASP.NET's session state module and session state data sources:

- InProcSessionStateStore, which stores session state in memory in the ASP.NET worker process
- OutOfProcSessionStateStore, which stores session state in memory in an external state server process
- **SqlSessionStateStore** , which stores session state in Microsoft SQL Server database

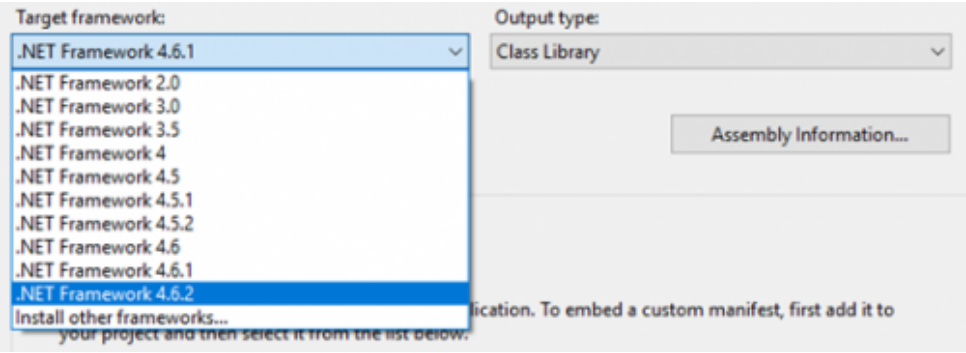
This blog post focuses on the **SqlSessionStateStore** provider and describes how you can configure it to use SQL Server In-Memory OLTP as the storage option for session data. You can either use the latest ASP.NET async version of the SQL Session State provider (which is the recommended approach), or configure an earlier version of the provider to work with In-Memory OLTP by downloading and running the In-Memory OLTP SQL scripts from our sql server samples github repo.

Option1: Use the latest ASP.NET async SQL Session State provider

The Microsoft ASP.NET team has released an async version of the session state provider that uses SQL Server as the data store and leverages async database operations to provide better scability. This version of the provider also includes built-in support for retry logic and works with both In-Memory and disk-bases tables.

Please follow the steps below to install the provider and configure it to use SQL Server In-Memory OLTP:

- **Step 1:** Install the latest Microsoft ASP.NET Async SqlSessionState Provider from Nuget. Note that the target framework of your web project need to be **4.6.2** (or above).



- **Step 2:** Open the project's Web.config file and add the following attributes in the SqlSessionStateProviderAsync element to enable In-Memory OLTP and to adjust the retry times and retry interval (in ms).
  - **UseInMemoryTable:** Set the value to true for In-Memory OLTP or false for disk-based tables
  - **MaxRetryNumber:** The maximum number of retries. Set to 0 if you want to disable retries.
  - **RetryInterval:** Time in ms for the retry interval.In addition to these three attributes, a connection string section needs to be

added in the Web.config, where the name of the connection string should be the same as the value of **connectionStringName** attribute of the SqlSessionStateProviderAsync provider. Below is a sample ASP.NET Web.config Session State section that uses SQL Server In-Memory OLTP with MaxRetryNumber="5" and RetryInterval="100ms"

```
<connectionStrings>
  <add name="DefaultConnection"
       providerName="System.Data.SqlClient"
       connectionString="Server=.;Database=SessionStateDB;Integrated Security=True"/>
</connectionStrings>
<system.web>
  <sessionState cookieless="false"
                regenerateExpiredSessionId="true"
                mode="Custom"
                customProvider="SqlSessionStateProviderAsync">
    <providers>
      <!--
          Please change the connection string named "DefaultConnection" to connect to an instance
          of SQL Server which you will use as the data store.
      -->
      <add name="SqlSessionStateProviderAsync"
           connectionStringName="DefaultConnection"
           UseInMemoryTable="true"
           MaxRetryNumber="5"
           RetryInterval="100"
           type="Microsoft.AspNet.SessionState.SqlSessionStateProviderAsync, Microsoft.AspNet.Sessior
    </providers>
  </sessionState>
```

Note: If the **UseInMemoryTable** , **RetryInterval** , and **MaxRetryNumber** attributes are not set, the provider will use regular disk-based SQL tables with the following default values for the RetryInterval and MaxRetryNumber:

SQL Server optionRetryInterval (ms)MaxRetryNumber

In-Memory OLTP	1	10
Disk based tables	1000	10

- **Step 3:** Set the database isolation level to **snapshot** by executing the following T-SQL command against the SQL database that you are using to store the session state. The Transactions with Memory-Optimized Tables article explains in detail why this is needed.

```
ALTER DATABASE CURRENT SET MEMORY_OPTIMIZED_ELEVATE_TO_SNAPSHOT = ON;
```

Option2: Use an earlier version of the ASP.NET Session State provider
In the case where you are not ready to upgrade to the latest async version of the SqlSessionStateStore provider, you can follow the steps below to configure an earlier version of the ASP.NET Session State provider to use In-Memory OLTP (with or without retry logic).

- **Step 1:** Download and run (on the target SQL Server to be used to store session state data) one of the following SQL Server In-Memory OLTP scripts: aspstate_sql2016 (no retry logic) or aspstate_sql2016 (with retry logic) . These scripts are based on work from early adopters that modified their SQL Server objects to take advantage of In-Memory OLTP for ASP.NET session state, with great success. To learn more, read the bwin.party case study Gaming site can scale to 250,000 requests per second and improve player experience .

  Based on your workload characteristics and the way your application handles session state you should decide:

  - **If retry logic is needed or not:** Choose the aspstate_sql2016 (with retry logic) if you need retry logic or the aspstate_sql2016 (no retry logic) if not. These scripts are required as the earlier versions of the ASP.NET Session State provider do not include retry logic or built-in support for In-Memory OLTP. You can read the Transactions with Memory-Optimized Tables article that explains the logic used to detect conflict and implement retry logic in the above T-SQL scripts.

- **If durability of both schema and data is required:** Currently, the two memory-optimized tables: dbo.ASPStateTempApplications and dbo.ASPStateTempSessions in both scripts are created with DURABILITY = SCHEMA_ONLY meaning that if SQL Server restarts, the table schema persists, but data in the table is lost. If durability of both schema and data is required, the script needs to be altered and the two tables above need to be created with: DURABILITY=SCHEMA_AND_DATA. The Defining Durability for Memory-Optimized Objects article explains the two durability options for memory-optimized tables in detail.

- **Step 2:** Set the Target framework of your web project to **.net 4.5** (or above).

- **Step 3:** Configure the web app to use the sql session state provider by modifying the Web.config as follows:
  • Set the mode attribute of the element to SQLServer to indicate that session state is stored in SQL Server.
  • Set the sqlConnectionString attribute to specify the connection string for SQL Server.

```
<sessionState
  mode="SQLServer"
  sqlConnectionString="data source=.;user id=USERNAME;password=PASSWORD"
  cookieless="false"
  timeout="20"
/>
```

Further Reading

• Microsoft ASP.NET Universal Providers
• In-Memory OLTP (In-Memory Optimization)
• Session State Provider
• Implementing a Session-State Store Provider

1 Comment

cb55555 Visitor

Apr 08 2019 11:41 A

@Perry Skountrianos Is there any known issue with using the SqlSessionStateProviderAsync with tables/stored procedures created with the SQL Server 2016 scripts mentioned above?

0 Likes

You must be a registered user to add a comment. If you've already registered, sign in. Otherwise, register and sign in.

Comment

What's new

- Surface Pro X
- Surface Laptop 3
- Surface Pro 7
- Windows 10 Apps
- Office apps

Microsoft Store

- Account profile
- Download Center
- Microsoft Store support
- Returns
- Order tracking
- Store locations
- Buy online, pick up in store
- In-store events

Education

- Microsoft in education
- Office for students
- Office for schools
- Deals for students and parents
- Microsoft Azure in education

Enterprise

- Azure
- AppSource
- Automotive
- Government
- Healthcare
- Manufacturing
- Financial Services
- Retail

Developer

- Microsoft Visual Studio
- Window Dev Center
- Developer Network
- TechNet
- Microsoft developer program
- Channel 9
- Office Dev Center
- Microsoft Garage

Company

- Careers
- About Microsoft
- Company News
- Privacy at Microsoft
- Investors
- Diversity and inclusion
- Accessibility
- Security

