HomeView

ORM DAR

Uniting Streaming Services on One Site

February 18, 2022

Unite
Christian Lam
Daniel Monge
Eric Truong (Team Leader)
Erina Lara
Michael Lamera

Introduction

Unite has considered three options for creating the data access layer:

- No ORM (Object Relational Mapping)
- Entity Framework (EF Core)
- Dapper

	No ORM	**Entity Framework Core**	**Dapper ORM**
Pros **0.50**	• No unnecessary overhead • Highly customizable and application specific +0.2	• Automatically produce objects and track changes on them • Full ORM • Translates CRUD operations from models into SQL operations +0.3	• Efficient and faster performance • Easily execute queries • Open source • Extensible to include a full ORM in the future • Manually making own SQL queries allows for easier error and query change management +0.5
Cons **-0.30**	• Higher possibility of running into implementation bugs or memory issues on implementation • May take up additional time to implement • Must write own SQL statements -0.3	• Larger framework, leads to performance issues • Not scalable -0.2	• Cannot generate class models • Cannot track objects and their changes • Must write own SQL queries -0.3
Subtotal 0.20	-0.10	+0.10	+0.20
Metrics			
Full ORM features 0.9	+0	+0.9	+0.7

Execution Speed (ms) 0.87	1.60 +0.40	1.47 +0.53	0.13 +0.87
Tracking (ms) 0.87	2.10 +0.10	2.00 +0.20	0.13 +0.87
Total (out of 2.84)	0.40	1.73	2.64

Conclusion

The team decided to proceed using Dapper's ORM for the data access layer. Dapper is considered a micro-ORM, meaning it does not have the full capabilities of a full ORM. This makes Dapper lightweight, allowing for it to thrive in scenarios where data is changed and requested frequently. Moreover, Dapper performs better in stateless applications, much like Homeview's web system. The micro-ORM allows the team to manually write the SQL queries, which allows greater control on what the database engine executes and grants the ability to tweak its performance. Additionally, a full ORM may not be able to override the generated SQL query, in the case a generated query provides undesirable performance results.

For the first calculated subtotal, our team decided to score each individual category's pros and cons. The highest possible number of points to gain from the pros would be +0.50, whereas the highest possible deduction of points from the cons would be -0.30. To calculate each option's subtotal from the recorded pros and cons, the number of cons would be subtracted from the option's total number of pros. In our case, the Dapper ORM scored the highest (by having +0.5 pros and -0.3 cons, resulting in a subtotal score of +0.20). Following the same procedure, the EF Core placed second with a subtotal of +0.10, whereas the No ORM option resulted last, with a subtotal of -0.10.

After calculating the subtotal, we analyzed the metrics of each option. For the Full ORM features scoring, the No ORM option scored +0 due to having no ORM features, whereas the EF Core option scored +0.90 for having full ORM functionality and features. On the other hand, the Dapper option scored +0.70 due to being a micro-ORM, with limited features and functionalities. For execution speed, the points for each option were scored based on speed, with the fastest speed scoring the most points. The points were calculated by subtracting the option's next whole

number (based on its speed) by the option's speed. For example, the Dapper ORM's execution speed was 0.13 ms. To calculate its metric score, the next whole number based on 0.13 would be 1.0, so we subtracted 1.0 by 0.13 to get a score of +0.87. As another example, the EF Core ORM's execution speed was 1.47 ms. To calculate its metric score, the next whole number based on 1.47 would be 2.0, so we subtracted 2.0 by 1.47 to get a score of +0.53. Likewise, for the Tracking metric, we followed the same scoring procedure for each of the ORM options. However, as a note, for the EF Core Tracking Scoring, due to being relative to the No ORM option's tracking speed, we estimated its scoring to be within the same range.

After calculating the subtotal and metrics from each ORM option, the highest possible scoring was a 2.84. The No ORM option scored the lowest, with a score of 0.40, with EF Core scoring 1.73, and the Dapper ORM option scoring 2.64. As a conclusion, based off its metrics and pros and cons, our team conclusively and unanimously decided to utilize the Dapper ORM.

References

https://exceptionnotfound.net/dapper-vs-entity-framework-core-query-performance-benchmarking-2019/