Efficiency Lab

Design Document

Isaac Styles

Friday February 17, 2017

ASP.NET Quiz Demo Design Document

This project was made in Visual Studio 2015 Update 3 using ASP.NET MVC 5. The .NET Framework is version 4.5.2. I used Entity Framework 6.1.3 for data management. The database must be seeded before first run. The init migration contains a schema and seed method that populates an instance of localdb and must be ran before the first run. Copy the following command and paste it in the Package Manager Console (PMC):

Update-Database -targetMigration 201702170645344\_init

The latest version of jQuery (3.1.1 at the time of submitting) is required for Steps and may be installed with Bower or the NuGet Package Manager. The only external package that is necessary is jQuery Steps. Although it seems to be included, the following command installs Steps via PMC:

Install-Package jQuery.Steps

The data model is a simple one-to-many join between Question and Answer. I used eager loading to populate all the properties at once as opposed to lazy or explicit loading. Eager loading yields the best performance for single-join queries (Dykstra & Anderson, 2016). The data model is designed to include any number of possible answers for each question. The constant Exam.NUMBER\_OF\_QUESTIONS determines how many questions are produced.

Shuffling the questions occurs in the ExamController. I chose the Fisher-Yates’ algorithm because it scales well and is generally regarded as having low bias. It is O(n) so it is appropriate for larger question-sets. This algorithm isn’t suitable for a casino-quality shuffle, but in small sets it will produce all combinations with equal probability (How not to shuffle - the Knuth Fisher-Yates algorithm, 2011).

The form posts a sparse Exam data structure contains a QuestionID and AnswerID for every question. The identifier of the given answer is returned in the Question.CorrectID property whose intended purpose is for the result of a server-side query. Because Questions.Answers isn’t initialized, the model returned by the form isn’t valid (as reflected in the ModelState.IsValid property). This conserves bandwidth because the Questions data structure posted to the server doesn’t contain an Answer. In order to prevent RadioButtonFor from selecting a default answer, Question.CorrectID is nullable. This doesn’t reflect the actual data type of CorrectID, which is required for every question. A Required data annotation ensures that no null values exist in the database even though the datatype is nullable.

jQuery Steps offers a Finished button once the last question is presented. One clicked, a JavaScript method performs input validation and, if appropriate, submits the form using AJAX. I disabled ASP.NET client-side validation and jQuery unobtrusive validation in favor of a JavaScript method to ensure an answer is selected for every question.

# References

Dykstra, T., & Anderson, R. (2016, October 14). *Reading Related Data*. Retrieved from EF Core with ASP.NET Core MVC Tutorial: https://docs.microsoft.com/en-us/aspnet/core/data/ef-mvc/read-related-data

*How not to shuffle - the Knuth Fisher-Yates algorithm*. (2011, July 17). Retrieved from I Programmer: http://www.i-programmer.info/programming/theory/2744-how-not-to-shuffle-the-kunth-fisher-yates-algorithm.html