Preface

I must admit this is somewhat of a scrappy implementation as I'm still getting to grips with how to use the MVC web framework effectively. I'm enjoying working more with it and learning as much as I can.

Assumptions

1. Building "an MVC application" refers to using the ASP.NET MVC Web Framework.
2. User's will separate their input integers with a comma.
3. Exporting sorts is done one by one. (Although it probably wouldn't be difficult to let a user do this all at once).
4. Exporting sorts to JSON can mean copying the JSON to the user's clipboard.
5. The user interface doesn't need to be pretty.

Decisions

1. I have used Entity Framework as the object relational mapper.

Storing Number Lists in a Database

This was the main issue I ran into. I spent some time considering it because it would impact the rest of the exercise at all levels.

Context:

My model of the number list (**`NumberList`**) relies on a generic List of integers (**`List<int> Numbers`**). This lets me use the **`List.Sort()`** method for easy sorting in ascending or descending order.

Problem

(AFAIK) Entity Framework does not support mapping collections of generic types to a database table because the generic type has no unique identifier AKA primary key.

Solution 1:

Make the list of integers into a list of custom objects (**`Number`**) which have a primary key and store an int. This can be made into a database table and each row will have a foreign key pointing to the **`NumberList`** it belongs to.

Pros

This should work with Entity Framework and still be straight forward to sort into ascending and descending order.

Cons

I have to worry about duplicate **`Number`**s in the database table which could clutter the database and impact overall performance. (Does this matter? *Probably!*)

I would need to configure some kind of one-to-many relationship between **`Number`** and **`NumberList`**.

Solution 2:

Represent the list of numbers as a string in the database and don't map the **`List<int> Numbers`** to the database at all.

Pros

No need for writing a class just to store an int.

No need to think about one-to-many relationships.

Should easily be able to convert into List<int> from string.

Cons

I was seriously considering this option but what if the list is e.g., millions of numbers long? Storing this in one field is the opposite way to use a relational database.

Solution 3:

Maybe storing the list as an array negates these problems?

Cons

I would likely have to implement a sorting algorithm (e.g. bubble sort).
(Which might be what the exercise is about to be fair…)

I've already decided to push ahead with Solution 1