CODE REVIEW

Repository with ViewModel (ASP.NET MVC)

0

I have db table Model.

Here is code:

```
public partial class Logging
{
    public Nullable<decimal> Latitude2 { get; set; }
    public Nullable<decimal> Longitude2 { get; set; }
    public int Speed { get; set; }
    public Nullable<int> Datatype { get; set; }
    public int Id { get; set; }

}
```

I want to calculate a difference

And decided to create View - Model

Here is code:

```
 public class HeatmapViewModel
{
    public int FirstStartDifference { get; set; }
    public int LastStartDifference { get; set; }
    public int coeff = 2;
    public int Difference;
}
```

And I decided to make calculations on Repository layer Here is code:

```
  public List<HeatmapViewModel> GetStops()
    {
        using (var ctx = new GoogleMapTutorialEntities())
        {
            List<HeatmapViewModel> items = new List<HeatmapViewModel>();

            #region firstitem_calculation
            var firstitem = ctx.Loggings.Where(x => x.Datatype ==
2).AsEnumerable().Select(
                x => new Logging
                 {
                     Longitude2 = x.Longitude2,
                     Latitude2 = x.Latitude2,
                     CurDateTime = x.CurDateTime

                 }).FirstOrDefault();

             var midnight = new DateTime(firstitem.CurDateTime.Year,
       items.Add( new HeatmapViewModel
         {
           FirstStartDifference = difference_after_midnight
         });
      #endregion

      #region lastitem_calculation
      var lastItem = ctx.Loggings.Where(x => x.Datatype ==
2).AsEnumerable().Select(
         x => new Logging
         {
           Longitude2 = x.Longitude2,
           Latitude2 = x.Latitude2,
           CurDateTime = x.CurDateTime

         }).LastOrDefault();
      var before_midnight = new DateTime(lastItem.CurDateTime.Year,
lastItem.CurDateTime.Month, lastItem.CurDateTime.Day, 23, 59, 00);
      TimeSpan difference_before = (before_midnight - lastItem.CurDateTime);
      var difference_before_midnight = (int)difference_before.TotalMinutes;

      items.Add(new HeatmapViewModel
      {
        LastStartDifference = difference_after_midnight
      });

      #endregion
  return items;
    }


  }
```

And call on Controller just method from repo.

Like this:

```
public JsonResult GetStops()
    {
      var stops = repo.GetStops();

      return Json(stops.ToArray(), JsonRequestBehavior.AllowGet);
```

I want to know if the code on the repository is okay, or can I improve it?

c# asp.net asp.net-mvc

edited Oct 31 '17 at 20:18
Sam Oneła
12.1k 6 25 83

asked Oct 31 '17 at 12:12
S.E
32 1 7

I assume the db table Model is one generated by the EF reverse engineering tool, hence the `Nullable<>` right? – mrdnk Oct 31 '17 at 19:20

I also assume the Property names are just what the table column names are - hence the use of 2 on the end of Property names. – mrdnk Oct 31 '17 at 19:23

3 Answers

1

✓

So here is my take on how I'd change it. It might be over the top if the application is very simple and light, however, its how I'd rewrite it.

ExampleController

The actual implementation here would need to be updated, however this is to demonstrate how to use a new Service layer and passing in the repository.

```
public class ExampleController
{
    public JsonResult GetStops()
    {

        var loggingRepository = new LoggingRepository();
        var stops = new HeatMapService(loggingRepository);

        var heatMapViewModel = stops.GetMyHeatmap();

        return Json(stops.ToArray(), JsonRequestBehavior.AllowGet);

    }
}
```

Firstly it's useful to have something to orchastrate the work, and so I'd add a HeatMapService.

```
public class HeatMapService
{

    private LoggingRepository LoggingRepository { get; set; }

    // Pass in the LoggingRepository
    public HeatMapService(LoggingRepository loggingRepository)
    {
        LoggingRepository = loggingRepository;
    }

    // Have a service method, whos job it is to return the HeatmapViewModel for your use
case
    public HeatmapViewModel GetMyHeatmap()
    {

        // We individually call the Repository for the first and last logs
        var firstLog = LoggingRepository.GetFirstLog();
        var lastLog = LoggingRepository.GetLasttLog();

        // We ask the entity to get the answer to the difference before and after
midnight
        var firstDifference = firstLog.GetDifferenceAfterMidnight();
        var lastDifference = lastLog.GetDifferenceBeforeMidnight();

        // Finally we create a new HeatmapViewModel - using a factory method
        return HeatmapViewModel.CreateHeatMapFromTwoDifference(firstDifference,
lastDifference);
    }

}
```

The Logging Model

Ask the model questions about the model

```
public class Logging
{
    // Id moved to the top, as this is assumed to be the unique id for this entity
    public int Id { get; set; }
    public string Imei { get; set; }
    public decimal? Latitude2 { get; set; }
    public decimal? Longitude2 { get; set; }
    public int Speed { get; set; }
    public int? Datatype { get; set; }

    // Get the difference after midnight, on the actual Model itself
    public int GetDifferenceAfterMidnight()
    {
        return (int)(this.CurDateTime - this.CurDateTime.Date).TotalMinutes;
    }

    // Get the difference before midnight, on the actual Model itself
    public int GetDifferenceBeforeMidnight()
    {
        return (int) (this.CurDateTime.Date.AddHours(23).AddMinutes(59) -
this.CurDateTime).TotalMinutes;
    }

}
```

HeatmapViewModel

Create a new instance of HeatmapViewModel using a factory method, restricting how the model can be created.

Also note that the Properties are private set - so can only be set via the factory method (or via additional non-static methods).

```
public class HeatmapViewModel
{
    public int FirstStartDifference { get; private set; }
    public int LastStartDifference { get; private set; }
    public int Coeff => 2;
    public int Difference { get; private set; }

    // Just pass in your two ints to create a new HeatMapModel
    public static HeatmapViewModel CreateHeatMapFromTwoDifference(int
firstStartDifference, int lastStartDifference)
    {
        return new HeatmapViewModel
        {
            FirstStartDifference = firstStartDifference,
            LastStartDifference = lastStartDifference
        };
    }
}
```

LoggingRepository

Simpify the repository methods, to just get the entries you need. Single Responsibilty - return individual log entries from the ctx

```
public class LoggingRepository {

    // Get the First Log
    public Logging GetFirstLog()
    {
        using (var ctx = new GoogleMapTutorialEntities())
        {

            var firstitem = ctx.Loggings.FirstOrDefault(x => x.Datatype == 2);
            return firstitem;
        }
    }

    // Get the Last Log
    public Logging GetLasttLog()
    {
        using (var ctx = new GoogleMapTutorialEntities())
        {

            var firstitem = ctx.Loggings.LastOrDefault(x => x.Datatype == 2);
            return firstitem;
        }
    }

}
```

edited Oct 31 '17 at 21:00

answered Oct 31 '17 at 19:29

mrdnk

126 5

The factory method, is probably over-the-top for this case, and a simple constructor would probably be better. – mrdnk Oct 31 '17 at 21:08

0

Some remarks on your code style.

- Don't needlessly abbreviate: why use `CurDateTime` when `CurentDateTime` would be much easier to read?
- I haven't use the `Nullable<T>` notation in a decade or so; these days I always use `int?` etc.
- Why is there a "2" at the end of `GPSDateTime2` , `Latitude2` , `Longitude2` ?
- GPSDateTime2 doesn't follow the naming rules:

  > Do capitalize only the first character of acronyms with three or more characters, except the first word of a camel-cased identifier.

- Public properties like `coeff` should be capitalized.
- Avoid regions. There's no need to divide a 40-odd method using regions.
- Don't do this: `var midnight = new DateTime(firstitem.CurDateTime.Year, firstitem.CurDateTime.Month, firstitem.CurDateTime.Day, 00, 00, 00);` That's why DateTime has the `Date` property.
- Don't compound words using underscores: `difference_after_midnight` , `before_midnight` , `difference_before` . C# isn't PHP.
- Don't use numbers as types: `Where(x => x.Datatype == 2)` . "2" means nothing. This is why enums exist.

answered Oct 31 '17 at 13:27

BCdotWEB

8,895 1 20 39

It's some kind of Legacy code. I don't named properties – S.E Oct 31 '17 at 13:28

I agree that regions are unwarranted inside a (well designed) method, but I disagree with the linked answer's notion that regions outside of methods are equally bad. Their justification hinges on adherence to StyleCop already enforcing some organization to the code, but completely forgets that StyleCop is in no way ubiquitous. Regions have valid use cases that have no influence on the quality of the code whatsoever. **Irresponsibly using regions to hide bad code** is not the same as regions allegedy being the **cause** of bad code. That's just ridiculous. – Flater Oct 31 '17 at 13:45

0

There is some duplication in the code which could be removed. Now granted, seeing some similar looking code twice doesn't always mean we *must* remove it, but we can at least have a look.

```
ctx.Loggings.Where(x => x.Datatype == 2).AsEnumerable().Select(
                        Latitude2 = x.Latitude2,
                        CurDateTime = x.CurDateTime

                      })
```

It seems this is performed twice, once with tacking on `FirstOrDefault` , another time with `LastOrDefault` . Extracting it and giving it a name might help.

answered Oct 31 '17 at 16:54

user1777136
209 1 6