Age Ranger

Solution description

AgeRanger is a world leading application designed to identify person's age group!

Used technologies:

- EF6 for SQLite
- Web Api
- AngularJs 1.6.4
- NUnit
- Moq
- StructureMap.MVC5 3.1.1
- TwitterBootstrap
- Elmah
- .NET Framework 4.5.2

Tools:

- SQLite compact toolbox https://marketplace.visualstudio.com/items?itemName=ErikEJ.SQLServerCompactSQLiteToolbox
- NUnit test adapter https://marketplace.visualstudio.com/items?itemName=NUnitDevelopers.NUnit3TestAdapter
- DB browser for SQLite http://sqlitebrowser.org/
- Visual Studio 2015 Express

Solution description:

Solution consists of 3 layers:

- Data layer is responsible of communication with Database, which is in our case SQLite database. This layer has it's own data models. All communication with this layer implemented via interfaces.
- Business layer is responsible of data processing. This layer has it's own business models.All communication with this layer implemented via interfaces.
- Presentation layer is responsible of preparing data for frontend.

Communication with frontend implemented via WebAPI.
Frontend part is implemented using AngularJS.
Markup done using TwitterBootstrap.

As far is it is mentioned in the task description that database administrator has in mind to migrate database to SQL Server, I used Dependency Injection container StructureMap to decouple layers. Business layer knows nothing about underlying data layer, it knows only about interfaces that data layer should provide. So if in the future we will have to migrate to SQL Server, we can implement new data layer for SQL Server and then change only two lines of code in the main solution (AgeRanger.BusinessLogic does not references AgeRanger.Data project):

```
For<IAgeGroupRepository>().Use<AgeGroupRepository>();
For<IPersonRepository>().Use<PersonRepository>();
```

Frontend is implemented as AngularJS application with call to asynchronous Web API. User can search Persons by First name, Last Name, Age and Age group using only one search input field.

Exception handling in AgeRanger application id implemented using:

- Elmah for MVC5 with small customization to make it work for Web API
- Custom Errors
- Error messages in frontend

User receives basic information about that something is happened and can then contact support so they will be able to check the exact reason by checking http://localhost:*****/elmah url, all exceptions that occur in application will be visible there. Elmah can be configured to be visible remotely too, if it is necessary.

Unit tests are implemented using NUnit and Moq frameworks.

To run solution you should do the next steps:

1. Clone repository locally
2. Open in Visual Studio 2015.
3. Run Restore NuGet packages command on context menu of the solution
4. Set web project as a startup project
5. Run.

Database file can be found in the folder App_Data of the Web project, so if you want to try your own database file, you can just replace it and rebuild the solution.