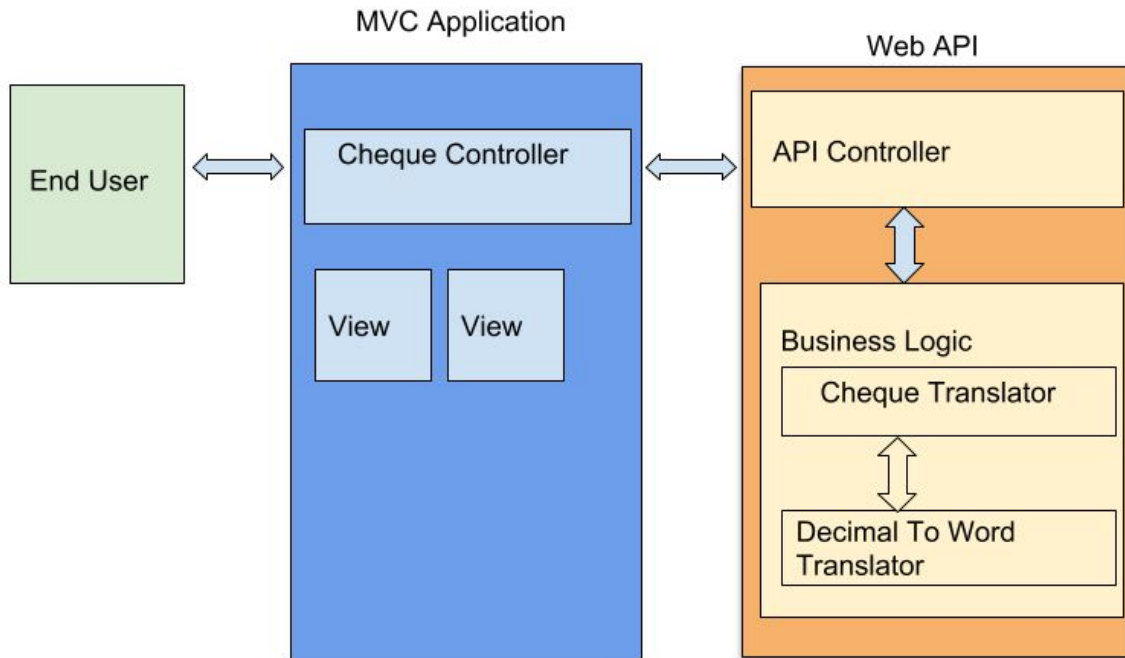


AKQA Cheque Translate technical description

Target Frameworks : Microsoft .NET Framework 4.61

Flow Diagram



Dependencies :

1. Ninject
2. Ninject.WebApi.DependencyResolver
3. Ninject.Web.Common
4. nunit.framework
5. NSubstitute

Technical Description (Project Structure):

1. **Cheque.Writing.WebAPI**
Web API application containing endpoint to translate the check into words
2. **Cheque.Writing.App.csproj**
MVC5 front end application utilizing "Cheque.Writing.WebAPI" project to get JSON response using REST call and display the translation in words
3. **Cheque.Writing.Entities.sproj**
Containing Entities utilized by Web API project transmit the database back to end user inform or JSON or XML depending on the request type
4. **Cheque.Writing.Common.csproj**

Common project containing common functionality such as “NuberstoWords” class used to convert int64 to words class and “Int64ToWordsExtension” Extension method class for Int64 to Words.

5. **Cheque.Writing.App.Business.csproj**

contains business logic to translate check in to words. IT contains concrete class “TranslateChequeToWords” having function to get Account holder name and amount of check.

6. **Cheque.Writing.WebAPI.Tests.csproj**

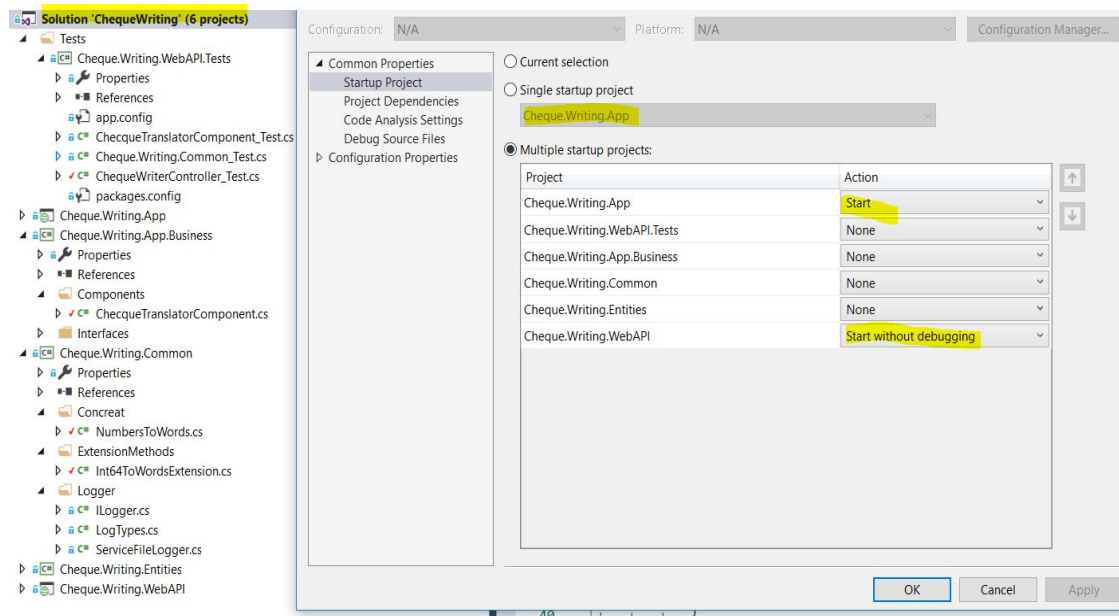
Contains Testing function for Web API controller , NumberstoWord and TranslateChecqueToWords classes

Programming techniques utilized :

- Inversion of control
- dependency injection using “Ninject.WebApi.DependencyResolver”
- For testing I used “nunit” and “NSubstitute”

How to Run:

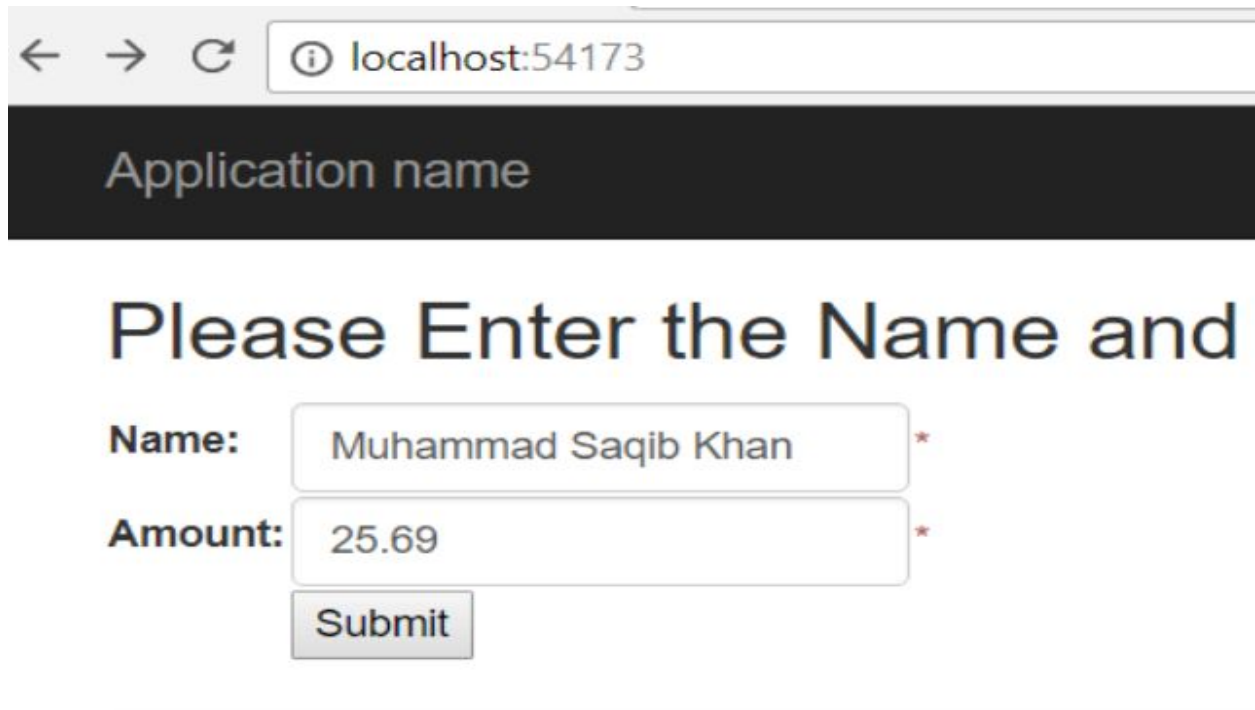
- 1) Enable project setting for both for Both WebAPI and MVC project to run in parallel as shown in the following figure



Note : Second option is to host your Web API(Cheque.Writing.WebAPI) application on IIS server and update BaseUrl (“baseUrl”) in web.config of “Cheque.Writing.App” project.

baseUrl Key : <add key=”baseUrl” value=”<http://localhost:54208/>”/>

- 2) Once you run both api and mvc application,provide input value in the following screen as shown in below image and press submit button.



A screenshot of a web browser window. The address bar shows 'localhost:54173'. The page has a dark header with the text 'Application name'. Below the header, the main content area has a title 'Please Enter the Name and'. There are two input fields: 'Name:' with the value 'Muhammad Saqib Khan' and 'Amount:' with the value '25.69'. Both fields have a red asterisk to their right. Below the input fields is a 'Submit' button.

← → ↻ ⓘ localhost:54173

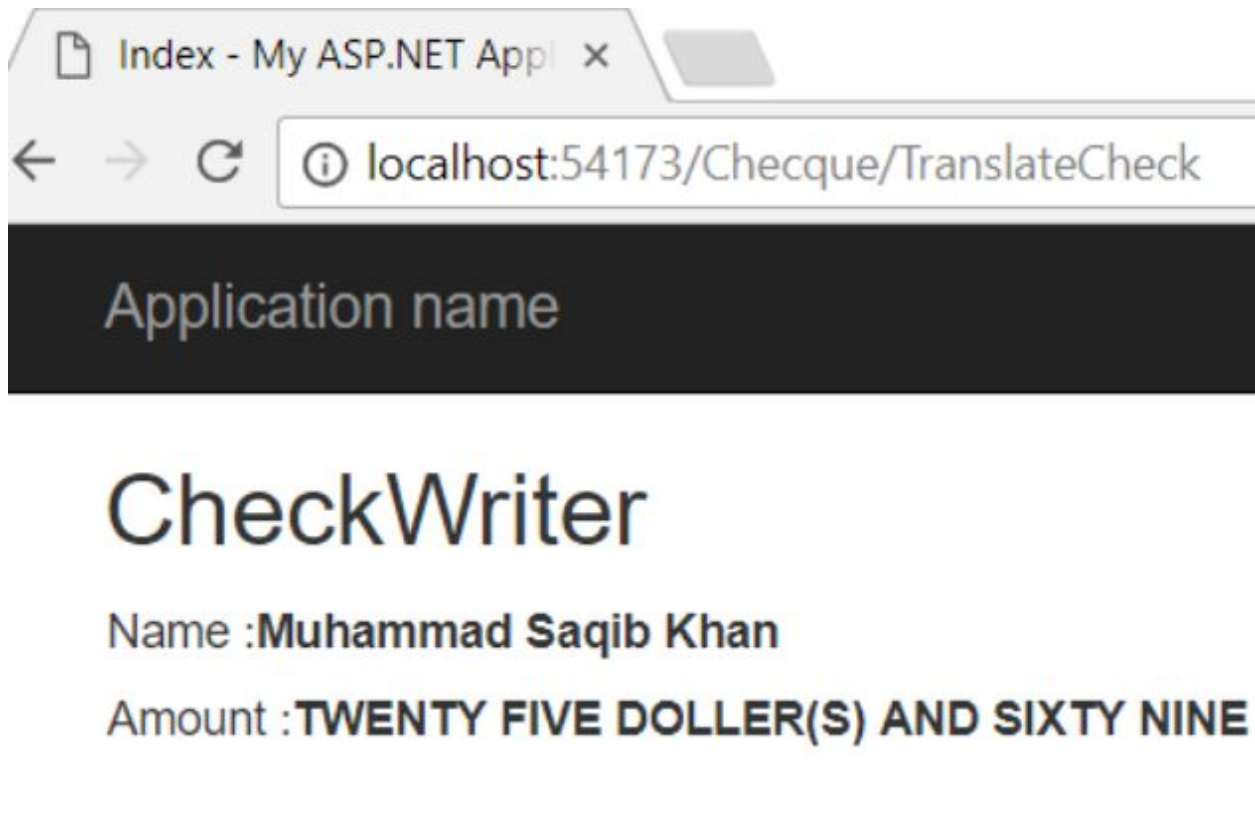
Application name

Please Enter the Name and

Name: *

Amount: *

3) You will see the following result.



A screenshot of a web browser window. The address bar shows 'localhost:54173/Checkue/TranslateCheck'. The page has a dark header with the text 'Application name'. Below the header, the main content area has a title 'CheckWriter'. There are two lines of text: 'Name :Muhammad Saqib Khan' and 'Amount :TWENTY FIVE DOLLER(S) AND SIXTY NINE'.

Index - My ASP.NET Appl ×

← → ↻ ⓘ localhost:54173/Checkue/TranslateCheck

Application name

CheckWriter

Name :Muhammad Saqib Khan

Amount :TWENTY FIVE DOLLER(S) AND SIXTY NINE