# Bank Core by danielanywhere

A Sample Bank Core Work in Progress Illustrating Multiple Techniques

## Introduction

This library is a sample work in progress that builds a tiny bank core using a fully-functional *zero programming* approach. This library consists of five separate projects.

* An Excel file named **BankWorksheet.xlsm**, available in the **Docs** folder.
* SQL Database backend. Available in the **SQL** folder.
* **BankViewModel**. MVVM Model and ViewModel support, initiated by a data-first Entity Framework approach in Microsoft Visual Studio 2013.
* **BankWEB**. MVVM UI using Kendo UI MVVM for JQuery through raw Web API 2 via REST and ASP.NET.
* **BankWPF**. MVVM desktop UI using Microsoft Windows Presentation Framework.

## Purpose

Simply, the point of this system is to construct software using only specifications and declarative syntax found in the Excel workbook. With this approach, an entire functional system is constructed from an easy to understand inventory made up of some component sheets roughly representing the tables, and short templates that roughly represent the shape of the code, all contained in well-organized spreadsheets.

The specific files in these projects have been created by running Excel macros to produce the following elements of the project.

* In **BankViewModel**, all the Collection-like and Item-like Observable classes that ride atop the Entity Framework to allow for full inline customization without any changes to Microsoft's wizard-based model.
  + **Account.cs**
  + **Branch.cs**
  + **Customer.cs**
  + **Employee.cs**
* In **BankWEB**, all the REST controllers for Web API 2 that service the objects and changes in transit.
  + **AccountsController.cs**
  + **BranchesController.cs**
  + **CustomersController.cs**
  + **EmployeesController.cs**
* In **BankWEB**, the complete content of Index.html, which drives a multi-modal Single Page Application using Kendo UI MVVM for JQuery, and raw JQuery update to Web API 2 via REST.

Following are the sheet names currently available in the Excel file.

* **Configuration**. Config values for the file.
* **TemplateObjectCollection**. The code template for creating the Entity Framework hosting Item and Colllection classes for the ViewModel layer.
* **TemplateObjectController**. The code template for creating the Web API 2 REST controllers for the ViewModel and Web View layers.
* **TemplateHTMLSinglePageApp**. The code template for creating the Index.html Single Page Web Application.
* **ComponentAccount**. Declaration inventory for creating Account Data, Model, ViewModel, and View objects.
* **ComponentBranch**. Declaration inventory for creating Branch Data, Model, ViewModel, and View objects.
* **ComponentCustomer**. Declaration inventory for creating Customer Data, Model, ViewModel, and View objects.
* **ComponentEmployee**. Declaration inventory for creating Employee Data, Model, ViewModel, and View objects.

## Integration and Operational Traits

This project demonstrates the successful integration of the following infrastructure and techniques.

* Data / JSON
* Data / SQL Server 2014
* Data / XML
* Desktop / Excel-Driven Development Automation (EDDA)
* Desktop / Microsoft VisualBasic for Applications (VBA6)
* Desktop / Visual Studio Projects
* Desktop / Windows Presentation Foundation (WPF)
* Programming / C#
* Programming / Class Libraries
* Programming / Dynamic Expression Handling (Runtime User Expressions)
* Programming / Entity Framework
* Programming / MVVM
* Scripting / JavaScript
* Scripting / JQuery
* Web / HTML5
* Web / KendoUI.MVVM

## Still TODO

As mentioned, this library isn't quite finished yet. Following is a short list of items to be completed for the sample to be considered 100%.

* **Evaluation**.
  + For evaluation purposes, publish the BankWEB project on the public server to allow guest use.
* **BankWorksheet.xlsm**.
  + Wire-up of additional control types and display formats according to the specified entries.
  + Include manual property override values into the field tag rendering sequence for properties on a component.
  + Define additional components.
  + Transfer SQL table creation and maintenance macros from another spreadsheet where it has already been tested, and link the macros to the Component\* sheet columns layout to allow for single point of definition, start to finish.
* **BankViewModel**.
  + Runtime expressions (ExpressionEvaluator.cs) have already been tested on the BankWPF project, but have not yet had any integration with components created by *zero programming*. Work toward the definition of a ComponentTransaction sheet, so runtime user-defined service charges can be demonstrated.
* **BankWEB**.
  + Add and integrate controllers and controls as more features are available.
* **BankWPF**.
  + The BankWPF project has not yet been templated, so create the basic Excel templates.